Security Analysis of Lightweight IoT Cipher
Chaskey

Dwivedi, Ashutosh Dhar

Document Version
Proof

Published in:
Cryptography

DOI:
10.3390/cryptography4030022

Publication date:
2020

License
CC BY

Citation for published version (APA):
Dwivedi, A. D. (2020). Security Analysis of Lightweight IoT Cipher: Chaskey. *Cryptography*, *4*(3), Article 22. https://doi.org/10.3390/cryptography4030022

cryptography

MDPI

Article

Security Analysis of Lightweight IoT Cipher: Chaskey

Ashutosh Dhar Dwivedi

Department of Applied Mathematics and Computer Science, Technical University of Denmark,
2800 Kongens Lyngby, Denmark; adhdw@dtu.dk or ashudhar7@gmail.com

Received: 3 July 2020; Accepted: 1 August 2020; Published: 5 August 2020

Abstract: This paper presents the differential cryptanalysis of ARX based cipher Chaskey using tree search based heuristic approach. ARX algorithms are suitable for resource-constrained devices such as IoT and very resistant to standard cryptanalysis such as linear or differential. To make a differential attack, it is important to make differential characteristics of the cipher. Finding differential characteristics in ARX is the most challenging task nowadays. Due to the bigger block size, it is infeasible to calculate lookup tables for non-linear components. Transition through the non-linear layer of cipher faces a huge state space problem. The problem of huge state space is a serious research topic in artificial intelligence (AI). The proposed heuristic tool use such methods inspired by Nested Tree-based sampling to find differential paths in ARX cipher and successfully applied to get a state of art results for differential cryptanalysis with a very fast and simpler framework. The algorithm can also be applied in different research areas in cryptanalysis where such huge state space is a problem.

Keywords: heuristic techniques; differential attacks; Chaskey cipher; ARX lightweight ciphers; nested tree search; single player games

1. Introduction

IoT has created new values by connecting network with various small devices, but security threat becomes more important issues in the recent reports of automobile hacking and illegal surveillance camera manipulation etc. In industry and academia alike, lightweight encryption has gained an enormous interest because of its simple operations and small size. Nowadays, IoT devices are required to use encryption to sensor devices with various restrictions. Some well established standard algorithm (e.g., AES) may not suitable for IoT as the basic requirements of these constrained devices are low power usage, low-cost hardware implementation, and latency. ARX stands for Addition/Rotation/XOR and is a family of lightweight symmetric-key encryption algorithms that are mostly designed with the very simple operations: Modular addition, bitwise rotation, exclusive-OR (XOR). ARX algorithms are generally secured against well-known attacks like linear and differential. The term ARX is very new and was introduced in 2009, but the concept of ARX is much older, and dates back to 1987—the FEAL cipher [1] used it first time. To analyze the security of symmetric algorithms, the most powerful tools are linear [2] and differential [3] cryptanalysis. However, for ARX ciphers there is not any proven security bound in the literature. ARX ciphers are very fast, and therefore designers use a large number of rounds to secure against these attacks. Finding an optimal differential characteristic (or differential path) is the most critical task to perform differential cryptanalysis. For ARX ciphers, finding differential characteristics is the most challenging task and involves months of manual calculations (as done by Wang et al. for several hash functions [4]) or to construct a heuristic search program. When applying differential cryptanalysis, one pays particular attention to non-linear operations such as an S-box or modular addition. The cryptanalysis of the substitution box (S-box) based algorithms are feasible in most of the cases. In the case of the S-P network such as the AES cipher, an S-box is typically 8- or 4-bit. Such a size allows computing the full difference distribution table (DDT) and investigating differential properties of the S-box and the algorithm. ARX-based designs use modular addition rather than S-boxes as a source of non-linearity. Word size in such ciphers are typically 32- or 64-bit and constructing a complete DDT is infeasible (it requires $2^{3n} \times 4$ bytes of memory for n-bit words). We face a huge number of possible difference transitions through modular addition box. Because of this, we need some efficient heuristic to circumvent this limitation. However, we have seen advancement in research to calculate a partial difference distribution table (pDDT) [5] to reduce the search space. But using such partial difference distribution table to find the differential path without any clever heuristic is still infeasible and requires several days to calculate differential characteristic. In artificial intelligence (AI) and in other areas such issues are very common where many problems have large searching space but no good heuristic available as a guide to find moves as the best path. In this paper, we developed a binary tree based random heuristic tool that improves results in each nested iteration. The algorithm tries to optimize the move at each level of the tree. For cryptanalysis purpose, we choose the Chaskey [6] cipher belongs to the ARX family. Chaskey cipher process a message m of 128-bit blocks and 128-bit key size K and very suitable algorithm for 32-bit micro-controllers.

2. Related Work

To our best knowledge, no differential cryptanalysis was performed against Chaskey cipher except authors of the cipher. However, few researchers applied a combined tool of differential-linear cryptanalysis and presented results of attack. In this paper, we mainly focus on differential cryptanalysis using tree search based heuristic tool and therefore only focus on differential cryptanalysis related articles for the given cipher. The article also focuses on the heuristic search tool, and therefore heuristic related analysis of ciphers are also important from the literature. In [6], authors applied differential cryptanalysis and found differential path for five rounds with probability 2^{-73}. The author also presented a differential path for eight rounds, but probability exceeds the exhaustive search bound.

In [7], the author applied differential-linear cryptanalysis and attacked six and seven rounds in the single-user setting. A differential-linear attack on round 7 takes 2^{78} data and time (respectively 2^{35} for six rounds). Authors also presented improved attack requires data complexity of 2^{48} and time complexity of 2^{67} (respectively 2^{25} data and 2^{29} time for six rounds). To improve the complexity of cryptanalysis, authors refine the partitioning technique proposed by Biham and Carmeli. In [8], authors performed some rotational cryptanalysis and produced result for full rounds with complexity 2^{86}. In [9–11], authors successfully performed differential cryptanalysis on ARX ciphers SPECK and LEA using heuristic inspired by tree search-based algorithms. Authors found a state of the art results for both ciphers.

In [12], authors proposed a heuristic tool that was capable of finding linear characteristics and also suitable for a relatively large state. The tool was designed for the primitives based on S-P networks. However, the design also allows extending the tool for other cryptographic primitives. Such a tool is important when designers of cipher design encryption algorithms and they can test the security margin of their cipher using this tool. The tool help designers to choose good S-Box and linear layer tool in an early designing process. As proof, they applied the presented tool on CAESAR candidates ICEPOLE, Ascon, Minalpher, Keyak and Prøst. However, this tool is not suitable for differential cryptanalysis of ARX ciphers.

3. Description of Chaskey

Chaskey cipher belongs to ARX family and designed jointly by Hitachi et al. [6] and COSIC research group. Chaskey is based on CBC-MAC and described as permutation-based design. The internal design of Chaskey follows the ARX construction; that is, operations for round functions are addition, rotation and XOR and therefore extremely fast on microcontrollers. It has a state size of 128-bit that consist of 4 32-bit words based on SipHash as shown in Figure 1. Initially, Chaskey has made for 8 rounds but later to increase the security margin of cipher, authors increased the number of rounds to 16. Chaskey is based on Even–Mansour structure that means there is no key schedule.

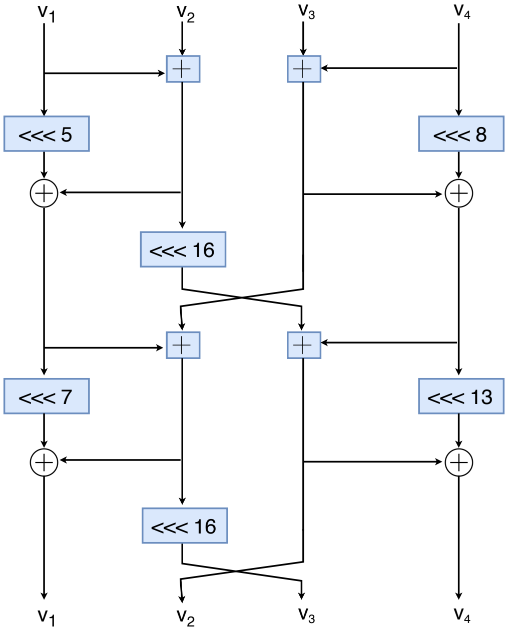


Figure 1. One round of the Chaskey permutation.

4. Differential Cryptanalysis

In this section, we present a short description of the cryptanalytic tool with respect to the n-bit block cipher. Iterated cipher consists of several numbers of similar round operations that are repeated to produce ciphertext for a given plaintext as input. In each round, a round key is required to mix with the round input. Differential cryptanalysis is the most important and powerful tool for analysing cryptographic primitives such as hash function or ciphers. Typically, it works in a chosen-plaintext scenario where an attacker can access encrypted ciphertext when providing plaintext chosen by him. For differential cryptanalysis, we chose pair of plaintext, and the pairs are related with each other by a constant difference; the difference can be defined by XOR operation or 2^n modular addition (see Figure 2). The attacker then computes the ciphertext difference hoping to detect some statistical difference in their distribution.

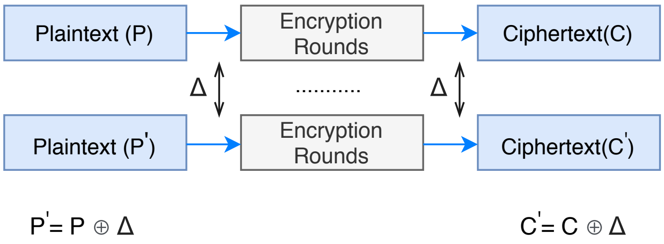


Figure 2. Difference propagation of plaintext pair.

For iterated block ciphers, encryption and decryption are defined by a composition of rounds $E_k = r_0 \circ r_1 \circ \cdots \circ r_k$. A differential characteristic Q (also called trail or path) is a sequence of differences through various rounds of the encryption. A sequence (see Figure 3) consist of an input difference Δ_0, followed by the output differences $\Delta_1, \Delta_2 \ldots \Delta_m$ of all the encryption rounds $(r_0, r_1, \ldots r_{m-1})$.

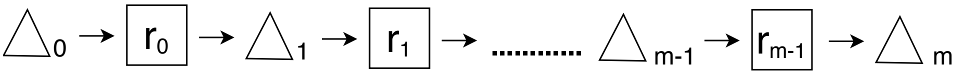


Figure 3. A differential characteristic over a sequence of rounds.

Each transition from Δ_i to Δ_{i+1} through the round r_i occurs with a certain probability. The total probability of differential characteristic is the product of all probabilities of these independent transitions through subsequent rounds.

When applying the differential cryptanalysis, one pays important attention to the non-linear component. Generally, for an input difference, there might be many possible output differences with different probabilities for a non-linear component, e.g., S-Box or modular addition. The size of S-boxes (see Figure 4) is typically 8- or 4 bit and therefore computing difference distribution table (DDT) is feasible.

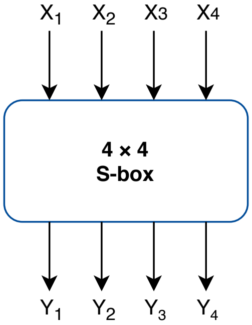
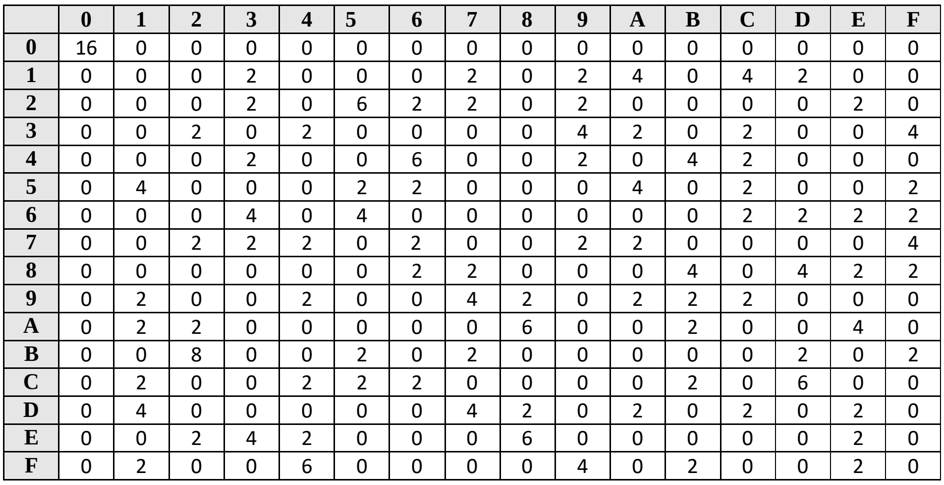


Figure 4. 4-bit S-box.

For example, the size of difference distribution table (see Figure 5) for a 4-bit S-box will be 2^8 (2^{16} for 8-bit respectively) where input size is 4 bit (8-bit for 8-bit S-box) and output size is 4 bit (8 bit for 8-bit S-box). The numbers inside the table can be used to calculate the probability of input-output through the non-linear layer.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Figure 5. Difference distribution table for 4-bit S-box.

However, when we talk about ARX ciphers, where the size of the non-linear component (modular addition) is generally 32-bit or 64-bit, it is infeasible to calculate DDT table (it requires $2^{3n} \times 4$ bytes of memory for n-bit words). In each round of Chaskey cipher, we face a huge number of possible difference transitions (see Figure 6) through modular addition box. This transition through the non-linear component is treated as a decision for the output with high probability and this is the place where we need a clever heuristic tool (due to unavailability of DDT table).

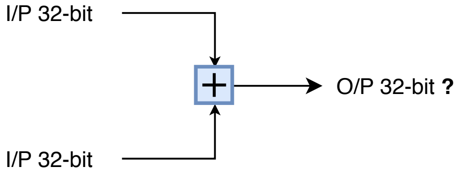


Figure 6. Transition through modular addition box.

In spite of searching most probable output with exhaustive search (2^{32} possible cases), the proposed nested algorithm tries to find high probability transitions through this non-linear layer using a heuristic approach. The algorithm randomly selects output and try to optimize the search with many iterations. However, to help the algorithm for fast results, we can reduce the search space by using partial difference distribution table (pDDT). Partial difference distribution table (pDDT) [5] does not contain full difference distribution table (that is practically infeasible) but contains only those XOR differentials $(a, b \rightarrow c)$ that has probability equal or greater than some threshold value p_{thres}.

$$(a, b, c) \in pDDT \Leftrightarrow DP(a, b \rightarrow c) \geq p_{thres}$$

However, by using a certain threshold, the algorithm can miss a few important paths with better results, but pDDT improves the algorithm speed in a good way. Variety of experiments can be performed at this level where threshold can be increased or decreased, and various results can be seen. In this work, we set the value of p_{thres} equal to 0.1 (see Table 1).

Table 1. The size of pDDT for 32- bit size with different thresholds.

Threshold Probability	Elements in pDDT
0.1	3,951,388
0.07	3,951,388
0.06	167,065,948
0.01	≥72,589,325,174

More details of pDDT can be found in the original paper. Transition with a higher probability has a low cost and vice-versa. The total cost can be found by multiplying the probabilities associated with each round transition.

5. Calculating Differential Probabilities

To calculate the XOR-differential probability of addition modulo 2^n with input differences p and q and output difference r, Moriai and Lipmaa [13] presented some formulas. Moriai and Lipmaa proved that the differential $(p, q \rightarrow r)$ is valid iff:

$$eq(p \lll 1, q \lll 1, r \lll 1) \wedge (p \oplus q \oplus r \oplus (q \lll 1)) = 0 \quad (1)$$

where

$$eq(s, t, u) := (\neg s \oplus t) \wedge (\neg s \oplus u) \quad (2)$$

For each differential that is valid $(p, q \rightarrow r)$, we define the weight $w(p, q \rightarrow r)$ of the differential as follows:

$$w(p, q \rightarrow r) = -\log_2(xdp^+(p, q \rightarrow r)) \quad (3)$$

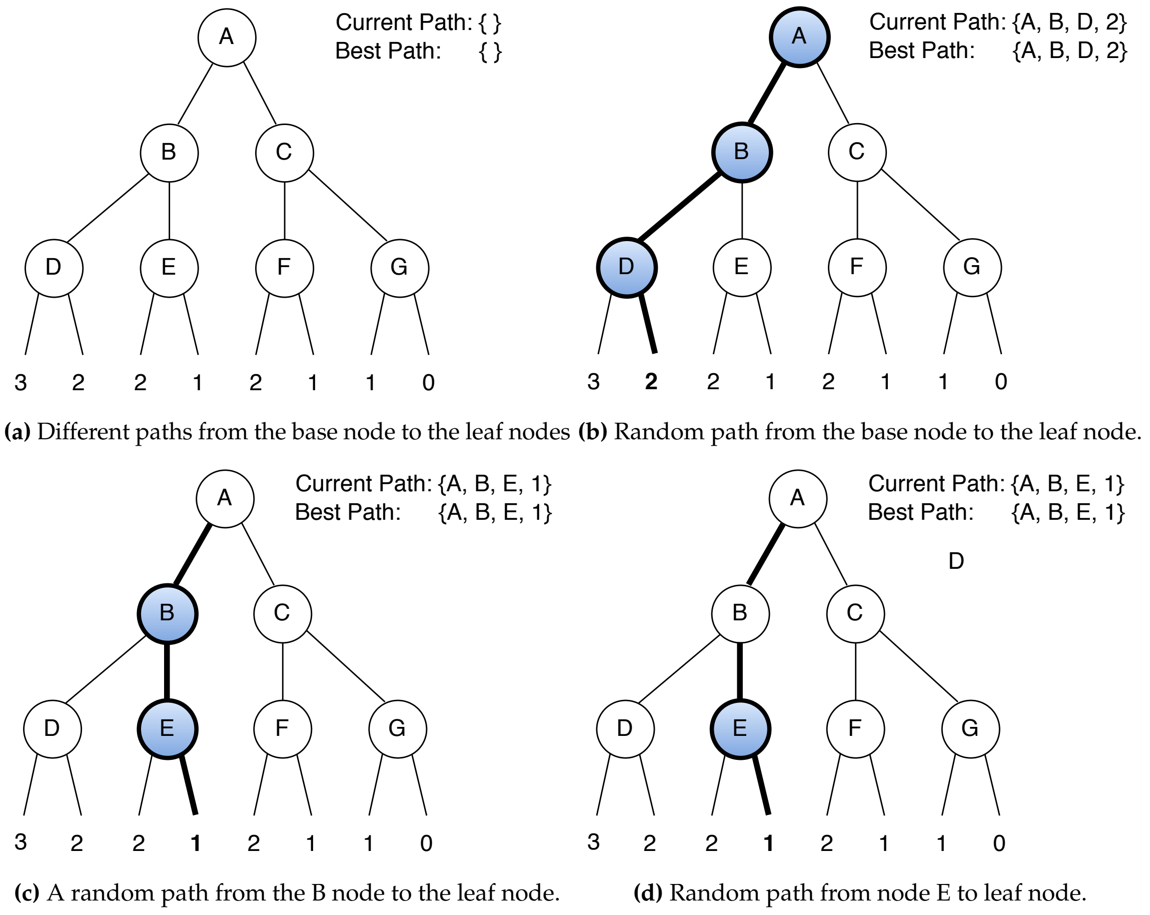
Valid differential weight can then be calculated as:

$$w(p, q \to r) := h^*(\neg eq(p, q \to r)), \tag{4}$$

where $h^*(l)$ denotes the number of non-zero bits in l, not counting $l[n-1]$.

6. Heuristic Tool Used to Find Differential Path

The proposed heuristic is a random sampling method based on binary tree-like structure. Such random sampling algorithms [14] are useful in the deep neural network where it is hard to formulate an evaluation function. Some researchers used nested tree search like methods to solve single-player games like 16 × 16 Sudoku and applied successfully to guide the search toward the best positions. To understand the heuristic tool, lets take an example of the tree-like structure (see Figure 7a). We represent all possible path in the form of a tree. The roots represent the initial points and leaves represents the ending point. Consider each left and right move increase the cost by 1 and 0, respectively. Our goal is to reduce the cost by using the proposed heuristic tool. The list *BestPath* and *CurrentPath* represents the best path from the previous search and random path currently under investigation, respectively. The last element in both lists represents the score of a random move.



(a) Different paths from the base node to the leaf nodes **(b)** Random path from the base node to the leaf node.

(c) A random path from the B node to the leaf node. **(d)** Random path from node E to leaf node.

Figure 7. Nested tree search.

Initially, the lists are empty, as shown in Figure 7a. Once we make a random move from root to the leaf, we fill the *CurrentPath* list. Initially, the *BestPath* list was empty, and therefore the current path will also become the best path (see Figure 7b). For random moves in the current path, the selected nodes are coloured with dark blue.

Next step is, we go one level down by following the *BestPath* list and start a random move from node *B*. This time our score is better than the previous one, and therefore, we update the best path with new nodes (see Figure 7c). We again go one step down by following the *BestPath* list and start a random move from *E*. However, this time we do not get better score than the previous and therefore do not change the *BestPath* list from *CurrentPath* list (see Figure 7d). Note that every time we go one level down by following the stored best path list. Now we reached at the end of the tree and therefore we again start from the root node for the next iteration. This way, we are moving toward better results.

7. Pseudocode of Heuristic Tool

The source of non-linearity in Chaskey is a modular addition where the algorithm needs to take a decision. The block of the cipher is divided into four parts v_1, v_2, v_3, v_4 with four modular addition operations in each round. We take four random values as the input of the algorithm. The algorithm can pick these input either from pDDT or randomly other values. In each round, algorithm initially check values from pDDT and if not found, it takes a random valid output and calculates the weight. Our goal is to search those paths for which weight is optimal. For simplicity of the algorithm, we skipped ciphers all encryption operations and only mentioned input-output and weight of non-linear components in the function.

RANDOM-PATH function (see Algorithm 1) has one input that provides current round position. Consider the cipher has $r = 5$ rounds, in such case, for the first time *current_round_position* will be 1 and function will run from round 1 to 5. Round is similar to node position in the tree-like structure and therefore when we go one level down in the tree, it means next round of cipher and therefore second time the function will run from round 2 to round 5. The state of the cipher will keep changing after each round of operations, and therefore for each loop input will be different than the previous one. If inputs belong to partial difference distribution table, then the output and weight are taken from the same table; otherwise, we calculate the weight for a valid differential output.

Algorithm 1 Random move to find differential path

```
function RANDOM-PATH($current\_round\_position$)
    while $current\_round\_position \neq last\_round$ do
        if $(input_1, input_2$ and $input_3, input_4) \in$ pDDT then
            output and weight is added to the path and weight list, respectively
            Similarly, do for other two modular operations
        else
            $output_1 = input_1 \oplus input_2$
            $weight_1 = weight(input_1, input_2, output_1)$
            $output_2 = input_3 \oplus input_4$
            $weight_2 = weight(input_3, input_4, output_2)$
            Add $weight_1, weight_2$ and $output_1, output_2$
            to the weight and path list, respectively
            Similarly, do for other two modular operations
        end if
    end while
return $path, weight$
end function
```

The recursive function NESTED-HEURISTIC (see Algorithm 2) call itself at each level of the tree. In our case, the function calls itself at each round, until it reaches the last round. In each and every call it updates the global variable *Best_weight* if it finds a better weight than the previously-stored best weight. Initially, the best weight is assigned as very big value, and the goal is to reduce it to optimal weight. The recursive function NESTED-HEURISTIC can be called any number of times until we get the optimal weight.

Algorithm 2 Recursive Nested Heuristic function

```
function NESTED-HEURISTIC($current\_round\_position$)
    while $current\_round\_position \neq last\_round$ do
        $temp\_path, temp\_weight = RANDOM - PATH(current\_round\_position)$
        if ($temp\_weight < best\_weight$) then
            $best\_weight = temp\_weight$
            $best\_path = temp\_path$
        end if
        Follow $best\_path$ and go to the next round
        $current\_round\_position = current\_round\_position + 1$
        if $current\_round\_position \neq last\_round$ then
            NESTED-HEURISTIC ($current\_round\_position$)
        end if
    end while
end function
```

8. Results

In this paper, we used the tree search based heuristic tool to find the differential path in ARX cipher Chaskey. The proposed algorithm is applied to round reduced Chaskey. For the size of the 128-bit state, it only make sense to analyse the path with probability higher than 2^{-128}. To make a meaningful attack, the algorithm should faster than an exhaustive search in the 128-bit state. We report the differential path for five rounds of the cipher with probability 2^{-103} (see Table 2). Instead of taking random values, we use pDDT table and set the threshold equal to 0.1 and selected only those paths that have a probability greater than 0.1. The number of values in search space that has a probability greater than 0.1 is 3,951,388. However, at this point algorithm have many options to change the value of the threshold, and it will change the results. In our analysis to find differential paths using the proposed heuristic, the time complexity of algorithm with n bit block is $\mathcal{O}(n^3)$ and therefore produce the result very fast. Note that the algorithm is based on random sampling and therefore, an observer can not expect the same result every time. To perform the experiment, instead of using any high processing server or cluster computers, we used normal PC, Mac OS, 2.3 GHz dual-core with 8 GB RAM. The code is written in python language and available at github [15].

Table 2. Differential trails for Chaskey Cipher.

Round	Block1	Block2	Block3	Block4	$\log_2 p$
1	0x00000008	0x00000008	0x00008181	0x00000081	−7
2	0x00000000	0x00000000	0x80000000	0x00000000	0
3	0x80000000	0x80000000	0x00008000	0x80001000	−9
4	0x80009810	0x80109080	0x90108000	0x92008082	−27
5	0x03964a02	0x0a008200	0x0a020213	0x1a428252	−60
				weight	−103

9. Conclusions

In this paper, we have analysed an ARX based cipher Chaskey with differential cryptanalysis. For ARX ciphers, finding differential characteristics is the most challenging task and involves months of manual calculations. The cipher has sufficient security margin against differential cryptanalysis but finding a differential path for round reduced Chaskey with limited time is one of the major contributions of this work (that we discussed in Section 8). The nested tree search tool can be applied to many cryptanalysis problems that do not have good heuristic to guide the search for an optimal path. With the given heuristic approach, we can perform many other experiments in future to find cryptanalysis results of various ciphers. We think it is essential to analyse these new, promising heuristics with a possibly wide range of ciphers and cryptanalytic tools. Our work helps to realize this goal.

Funding: This work is supported by a grant from the Independent Research Fund Denmark for Technology and Production, grant no. 8022-00348A.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Shimizu, A.; Miyaguchi, S. Fast Data Encipherment Algorithm FEAL. In Proceedings of the Advances in Cryptology—EUROCRYPT'87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, 13–15 April 1987; Chaum, D., Price, W.L., Eds.; Springer: Berlin/Heidelberg, Germany, 1987; Volume 304, pp. 267–278. [CrossRef]
2. Matsui, M.; Yamagishi, A. A New Method for Known Plaintext Attack of FEAL Cipher. In Proceedings of the Advances in Cryptology—EUROCRYPT'92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, 24–28 May 1992; Rueppel, R.A., Ed.; Springer: Berlin/Heidelberg, Germany, 1992; Volume 658, pp. 81–91. [CrossRef]
3. Biham, E.; Shamir, A. Differential Cryptanalysis of DES-like Cryptosystems. In Proceedings of the Advances in Cryptology—CRYPTO'90, 10th Annual International Cryptology Conference, Santa Barbara, CA, USA, 11–15 August 1990; pp. 2–21. [CrossRef]
4. Wang, X.; Yu, H. How to Break MD5 and Other Hash Functions. In Proceedings of the Advances in Cryptology—EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; Cramer, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494, pp. 19–35. [CrossRef]
5. Biryukov, A.; Velichkov, V. Automatic Search for Differential Trails in ARX Ciphers. In Proceedings of the Topics in Cryptology—CT-RSA 2014, San Francisco, CA, USA, 25–28 February 2014; Benaloh, J., Ed.; Springer International Publishing: Cham, Switzerland, 2014; pp. 227–250.
6. Mouha, N.; Mennink, B.; Van Herrewege, A.; Watanabe, D.; Preneel, B.; Verbauwhede, I. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In Proceedings of the Selected Areas in Cryptography—SAC 2014, Montreal, QC, Canada, 14–15 August 2014; Joux, A., Youssef, A., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 306–323.
7. Leurent, G. Improved Differential-Linear Cryptanalysis of 7-Round Chaskey with Partitioning. In Proceedings of the Advances in Cryptology—EUROCRYPT 2016, Vienna, Austria, 8–12 May 2016; Fischlin, M., Coron, J.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 344–371.
8. Kraleva, L.; Ashur, T.; Rijmen, V. Rotational Cryptanalysis on MAC Algorithm Chaskey. *IACR Cryptol. ePrint Arch.* **2020**, *2020*, 538.
9. Dwivedi, A.D.; Morawiecki, P.; Srivastava, G. Differential Cryptanalysis of Round-Reduced SPECK Suitable for Internet of Things Devices. *IEEE Access* **2019**, *7*, 16476–16486. [CrossRef]
10. Dhar, D.A.; Morawiecki, P.; Wójtowicz, S. Finding Differential Paths in ARX Ciphers through Nested Monte-Carlo Search. *Int. J. Electron. Telecommun.* **2018**, *64*, 147–150. [CrossRef]
11. Dwivedi, A.D.; Srivastava, G. Differential Cryptanalysis of Round-Reduced LEA. *IEEE Access* **2018**, *6*, 79105–79113. [CrossRef]
12. Dobraunig, C.; Eichlseder, M.; Mendel, F. Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates. In Proceedings of the Advances in Cryptology—ASIACRYPT 2015, Auckland, New Zealand, 29 November–3 December 2015; Iwata, T., Cheon, J.H., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 490–509.
13. Lipmaa, H.; Moriai, S. Efficient Algorithms for Computing Differential Properties of Addition. In *Fast Software Encryption*; Matsui, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 336–350.

14. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
15. Dwivedi, A.D. Security Analysis of Lightweight IoT Cipher: Chaskey. Available online: https://github.com/ashudhar7/Chaskeydifferential (accessed on 4 August 2020).