On Proving Security against Differential Cryptanalysis

Nicky Mouha

National Institute of Standards and Technology, Gaithersburg, MD, USA
nicky@mouha.be

Abstract. This submission to the "Conference for Failed Approaches and Insightful Losses (CFAIL)" revisits ePrint 2013/328. In its original version, this ePrint paper professed a proof that the Salsa20 stream cipher resists differential cryptanalysis. Subsequently, the paper was updated to point out an incorrect assumption in the proof. The title was also updated to start with the word "towards," which is standard cryptographer's speak for a failed approach. Despite the fact that the bug was never fixed and the manuscript was never published, it nevertheless went on to become one of the standard techniques to evaluate the resistance of a cipher against differential cryptanalysis. We will not only give insight into where the approach went wrong, but also explain why it is not easy to fix. Our goal is to give a new understanding of the value and limitations of proofs and experimental results, and suggest some lessons learned for both designers and cryptanalysts.

Keywords: CFAIL, Differential Cryptanalysis, ARX, Salsa20, SAT/SMT Solver

Foreword. *The research environment has an impact on the results that are published. Typical papers may require about three to six months of work, and should be ambitious but reasonable. This story is different. We set out to attain a seemingly impossible goal, fully aware that the road is bumpy and full of pitfalls. But after more than a year of painstaking effort, it appears that we somehow finally got there. Let's write up the results, put the paper on ePrint, and submit it to a reputable conference.*

The review comments may not be surprising: very laudable, but also very skeptical. But perhaps not too difficult to address: a few months of extra work should get us enough experimental support to convince the reviewers that the assumptions in the paper are correct. And so, we set up a large number of experiments, and one by one, they bolster our belief that the sky is clear, and that no clouds are to be found.

The last experiment on the list seems like a mere formality. But it isn't: the very last experiment actually shows that some of the main assumptions are incorrect. What do we do now? Attempts to fix the problem come up short, and the deadline to resubmit to another conference is looming... It seems that the only honest option left, is to admit the problem and describe it as clearly as possible. Let's update the ePrint paper and resubmit. The review comments, unsurprisingly, are all negative: this was a failed approach...

1 Introduction

For symmetric-key cryptographic algorithms, a basic design criterion is security against linear [22] and differential [6] cryptanalysis. In its most basic form, differential cryptanalysis is related to the question: "If we flip some bits at the input of the algorithm, what happens to the bits at the output?"

In the case of, let's say, a block cipher with a secret key K (drawn uniformly at random), if we can somehow "predict" changes in the ciphertext C when we flip some bits in the plaintext P, this would violate one of the most fundamental security assumptions: given any number of plaintexts and their corresponding ciphertexts, it should not be possible[1] to come up with the ciphertext corresponding to a "new" plaintext.

When this block cipher is used in a simple authentication scheme where the "challenge" is the plaintext, and the "response" is the corresponding ciphertext (encrypted under the secret key K), the security of the scheme crucially depends on this property: if the challenge is "fresh" (never used before), then the response should be unknown to anyone who doesn't possess K.

Despite the fact that security against differential cryptanalysis is so important, it may seem surprising that differential cryptanalysis is often one of the most effective attacks against symmetric-key cryptographic algorithms. Focusing, without much loss of generality, on the case of a block cipher, we see that the problem is twofold:

- For a b-bit block size, the number of non-zero plaintext differences is $2^b - 1$, which makes it clearly infeasible to explore all differences during the design phase (e.g., $b = 128$ in the case of the Advanced Encryption Standard (AES) [29]). The designer may therefore explore only a subset of these differences (e.g., differences in only one bit, or differences of a specific structure). But this often leaves the block cipher open to attacks by other "cleverly chosen" plaintext differences.
- For a specific plaintext difference $\Delta P = P \oplus P'$, it's often not a straightforward task to determine if there's a ciphertext difference that occurs more often than we would expect for an ideal block cipher. We will need, on average, 2^a pairs of plaintexts with difference ΔP to find a cipertext difference that occurs with a probability of 2^{-a} or higher, which is infeasible to do experimentally even for relatively small values of a. Designers will therefore use some theoretical estimates based on certain assumptions, but there is a discrepancy when the assumptions do not hold in practice. Also, the probabilities often depends on the secret key K, which further complicates the analysis.

To illustrate the point, we give examples of two well-known algorithms with vulnerabilities to differential cryptanalysis:

[1] More correct would be to say "infeasible": for example, any adversary can simply guess the secret key K and succeed. It is assumed that adversaries are computationally limited, and that such attacks can only succeed with a negligible probability.

TEA. Wheeler and Needham's Tiny Encryption Algorithm (TEA) [35] is a block cipher with a 128-bit key K. This key K consists of 32-bit subkeys K_0, K_1, K_2 and K_3. As Kelsey et al. [19] point out, we can flip the most significant bit (MSB) of both K_0 and K_1, without affecting the encryption algorithm. The same observation holds for the MSBs of both K_2 and K_3, which can be flipped without any effect. Note that this effectively reduces the key size by two bits: for every key, there are three other equivalent keys. This lead to an attack on Microsoft's Xbox gaming console, where TEA was used as a hash function [33].

MD5. Rivest's Message Digest 5 (MD5) hash function uses the Merkle-Damgård paradigm [14, 24], which turn a collision-resistant compression function into a collision-resistant hash function. The compression function of MD5 consists of four rounds, each round consisting of 16 steps. The step function of MD5 is almost identical to that of its predecessor MD4. One major change that was introduced, is that each step now adds in the result of the previous step. As shown by den Boer and Bosselaers [7], this significantly weakens the compression function of MD5 against differential cryptanalysis. In an average of four minutes on a 33 MHz 80386-based computer, they exploit this property to find a collision for MD5's compression function, thereby violating the Merkle-Damgård design principle.

2 The Wide Trail Design Strategy and AES

The aforementioned vulnerabilities of TEA and MD5 are obvious when we see them, and easy to fix if the designers knew about them. Hindsight is a wonderful thing, but what we really need is a systematic approach to avoid these problems. The Wide Trail Design strategy revolutionized the field by providing such an approach, and was used to design the AES block cipher.

Before we explain the Wide Trail Design Strategy, we introduce some terminology:

- A *difference* ΔX is the subtraction of two values: $\Delta X = X' - X$. We will assume that the subtraction is performed in a finite field with 2^n elements, so that subtraction is the same as an exclusive OR (XOR). This difference is therefore also known as an *XOR difference*.
- A *differential* $(\Delta X, \Delta Y)$ is an ordered pair consisting of an input difference ΔX and an output difference ΔY.
- The *differential probability* is the probability that the differential holds for a given cipher. Unless explicitly mentioned otherwise, this probability is assumes that inputs (X, X') and the key K are drawn uniformly at random, while satisfying $X \oplus X' = \Delta X$.
- A *trail* (also known as a *characteristic* or a *path*) is tuple of differences $(\Delta X, \Delta X_2, \ldots, \Delta X_{p-1}, \Delta Y)$.
- The *trail probability* is the probability that a trail holds for a given cipher, that is, the probability that not only the differential $(\Delta X, \Delta Y)$ holds, but

also the differences ΔX_2, ... ΔX_{p-1} for certain intermediate values that appear in the cipher.
- The *trail weight* (resp. *differential weight*) is the negative of the base-2 logarithm of the trail probability (resp. of the differential probability). Note that multiplying probabilities corresponds to adding weights.
- A trail (resp. differential) is *valid* if the trail probability (resp. differential probability) is non-zero.

The Wide Trail Design Strategy, when applied to AES, goes as follows. AES is an iterated block cipher, which consists of a round function that is iterated 10, 12 or 14 times, depending on the key size (resp. 128, 192 or 256 bits). Each round consists of the following steps: an addition of a 128-bit subkey (derived from the key), an application of sixteen 8-to-8-bit lookup tables (called S-boxes), and a linear transformation.[2]

The S-box of AES has been designed to optimally resist differental cryptanalysis [30]: for any non-zero differential of the S-box, the weight is at least 6.[3] In any trail, we will say that an S-box is *active* if it has a non-zero difference at its input. The Wide Trail Design Strategy of AES guarantees that any four-round trail has at least 25 active S-boxes [13].

Assuming that the trail probability can be accurately estimated by multiplying the probabilities of the differentials of every S-box, this means that every four-round AES trail has a probability of at most $2^{-25 \cdot 6} = 2^{-150}$, corresponding to a trail weight of at least 150. The subkeys help to justify this assumption: adding a random subkey after every round is effectively *whitening* the outputs of every round, i.e., ensuring that the outputs pairs are uniformly distributed.

The lowest differential weight may be less than the lowest trail weight: this effect is called *trail clustering* and happens when several trails have the same input and output difference. We can analyze this effect, and confirm that although there is some trail clustering for AES, the weight of the best trail still gives a reasonable indication of the weight of the best differential [12].

3 Differential Cryptanalysis for ARX Ciphers

An increasing number of ciphers are built using just three operations: addition modulo 2^n, bitwise rotation, and XOR. Collectively, these are known as Addition-Rotation-XOR (ARX) ciphers. Although ARX ciphers have a very fast performance in software, they cannot rely on a framework such as the Wide

[2] The last round is slightly different to allow for a more efficient hardware implementation, but we will not get into the details because it does not affect the security against differential cryptanalysis. To avoid that an adversary can trivially undo the last round, there is an additional subkey addition at very end of the cipher.

[3] It was long thought that no 8-to-8-bit S-box exists where every differential has a weight of at least 7. However, Dillon [9] has cast some doubt on this by constructing a 6-to-6-bit S-box where every differential has a weight of at least 5. Constructing a better 8-to-8-bit S-box against differential cryptanalysis, or proving that none exist, is still an open problem.

Trail Design Strategy to analyze their security against differential cryptanalysis. Until quite recently, their security has not been well-understood. For example, even the SHA-3 finalist Skein [15] was designed based on the "pre-differential-cryptanalyis" notion of *full diffusion*, meaning that every input bit affects every output bit. As we explained in Sect. 1, full diffusion is insufficient to claim resistance against differential cryptanalysis.

The bitwise rotation and XOR operations on n-bit words are linear operations in a finite field of 2^n elements, and therefore bit differences propagate with probability one. But this is not the case for the addition modulo 2^n. It was not until 2001 that Lipmaa and Moriai [21] studied the differential probability of just one addition operation.

Let $\mathrm{xdp}^+(\alpha, \beta \to \gamma)$ be the XOR-differential probability of addition modulo 2^n, with input differences α and β and output difference γ. Lipmaa and Moriai proved that the differential $(\alpha, \beta \to \gamma)$ is valid if and only if:

$$\mathtt{eq}(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0^n \ , \tag{1}$$

where

$$\mathtt{eq}(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z) \ , \tag{2}$$

and 0^n is an n-bit all-zero bit string.

For every valid differential $(\alpha, \beta \to \gamma)$, let us define the weight $w(\alpha, \beta \to \gamma)$ of the differential as follows:

$$w(\alpha, \beta \to \gamma) := -\log_2(\mathrm{xdp}^+(\alpha, \beta \to \gamma)) \ . \tag{3}$$

Lipmaa and Moriai proved that the weight of a valid differential can then be calculated as:

$$w(\alpha, \beta \to \gamma) := h^*(\neg\mathtt{eq}(\alpha, \beta, \gamma)) \ , \tag{4}$$

where $h^*(x)$ denotes the number of non-zero bits in x, not including the MSB.

Easy enough, right? So given a trail, can't we just add the weights of all the modular additions, and use this to estimate the weight of the trail? Not so fast! Mouha et al. point out in [27] that such an approach can quickly run into problems.

More specifically, they recall a result by Hong et al. [18] on the differential cryptanalysis of XTEA. Let us use $(i, j, ...)$ to denote an XOR difference of $2^i \oplus 2^j \oplus ...$. Hong et al. consider a three-round differential trail $(\delta, 0) \to (\delta, 0)$ where $\delta = (31, 22, 13, 4)$. In the third round of the trail, there are two consecutive additions modulo 2^{32}, as shown in Fig. 1.

Using Lipmaa and Moriai's formula for each addition, we find $w(\delta, 0 \to \delta) = 3$ and $w(\delta, \delta \to 0) = 3$. Simply adding the two weights would lead to a weight of $3 + 3 = 6$, but Hong et al. noticed some trail clustering, and estimated the weight to be 4.755. Mouha et al. [27] devised a method to accurately calculate the exact weight of differentials of a three-input addition, only to realize that the commutative property of the addition would have given the result immediately for Hong et al.'s XTEA trail: $w(\delta, \delta \to 0) + w(0, 0 \to 0) = 3 + 0 = 3$.

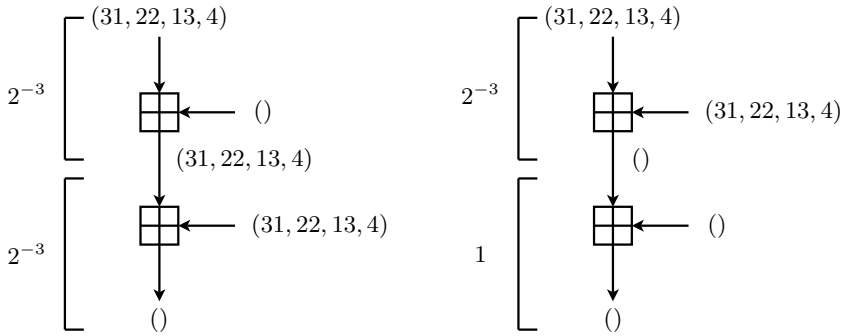

Fig. 1. A differential involving two additions modulo 2^{32} that appears in a trail by Hong et al. [18] for three-round XTEA. As pointed out by Mouha et al. [27], assuming that the differential probabilities are independent and therefore multiplying the differential probabilities of each addition, gives an incorrect probability of $2^{-3} \cdot 2^{-3} = 2^{-6}$ (shown left). Using the commutative property of the addition, we get the correct differential probability $2^{-3} \cdot 1 = 2^{-3}$ (shown right). The notation $(i, j, ...)$ denotes an XOR difference of $2^i \oplus 2^j \oplus ...$.

Rather than abandoning the approach of adding the weights of all the additions to estimate the trail weight, ePrint 2013/328 [26] argues that the problem might just be the structure of the ARX cipher, rather than the approach. They point out that the Salsa20 stream cipher [5] has an interesting property: unlike other ARX ciphers such as TEA, XTEA and Skein, the output of an addition is never directly used as the input of another addition.[4] This might give us some hope that Salsa20 avoids the aforementioned pitfall. But before we proceed, we should first give a description of Salsa20.

Although the following sections are relatively self-contained, the explanation will get a bit technical: our goal is to show that the shortcoming of the approach is not at all obvious, and not captured by any of the existing techniques in literature. Nevertheless, we will do our best to help the reader through the following sections, and hope that any lost readers will rejoin us in Sect. 8 for a philosophical discussion.

4 Description of Salsa20

Salsa20 is a stream cipher designed by Bernstein [5]. The originally proposed Salsa20 consists of $R = 20$ rounds. Later, Bernstein proposed two reduced-round variants: Salsa20/8 and Salsa20/12, consisting of 8 and 12 rounds respectively [3]. For the sake of clarity, the 20-round Salsa20 is sometimes referred to as Salsa20/20.

[4] To the best of our knowledge, LEA [17] is the other ARX cipher that has this property.

The Salsa20 stream cipher was submitted to the ECRYPT eSTREAM competition. At the end of the competition, the Salsa20/12 cipher was included in the eSTREAM portfolio. Although an attack was shown on Salsa20/8 [1], there are currently no known attacks on either Salsa20/12 or Salsa20/20.

Salsa20 supports both 128-bit and 256-bit keys. When using 128-bit keys, we first double the key to form a 256-bit key. Salsa20 operates on 32-bit words. The Salsa20 core function converts a 256-bit key $(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$, a 64-bit counter (t_0, t_1), a 64-bit nonce (v_0, v_1), and four 32-bit constants (c_0, c_1, c_2, c_3) into a 512-bit output. The inputs are mapped to a two-dimensional square matrix as follows:

$$\begin{bmatrix} x_0^0 & x_1^0 & x_2^0 & x_3^0 \\ x_4^0 & x_5^0 & x_6^0 & x_7^0 \\ x_8^0 & x_9^0 & x_{10}^0 & x_{11}^0 \\ x_{12}^0 & x_{13}^0 & x_{14}^0 & x_{15}^0 \end{bmatrix} \leftarrow \begin{bmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{bmatrix} \quad . \tag{5}$$

A Salsa20 round consists of four parallel `quarterround` functions, defined in Fig. 2:

$$(y_0^r, y_4^r, y_8^r, y_{12}^r) \leftarrow \texttt{quarterround}(x_0^r, x_4^r, x_8^r, x_{12}^r) \ , \tag{6}$$
$$(y_5^r, y_9^r, y_{13}^r, y_1^r) \leftarrow \texttt{quarterround}(x_5^r, x_9^r, x_{13}^r, x_1^r) \ , \tag{7}$$
$$(y_{10}^r, y_{14}^r, y_2^r, y_6^r) \leftarrow \texttt{quarterround}(x_{10}^r, x_{14}^r, x_2^r, x_6^r) \ , \tag{8}$$
$$(y_{15}^r, y_3^r, y_7^r, y_{11}^r) \leftarrow \texttt{quarterround}(x_{15}^r, x_3^r, x_7^r, x_{11}^r) \ , \tag{9}$$

followed by a matrix transposition:

$$\forall i, j : 0 \leq i < 4, 0 \leq j < 4 : x_{4i+j}^{r+1} \leftarrow y_{4j+i}^r \ . \tag{10}$$

After R rounds, the output is calculated by a feed-forward operation:

$$\forall i : 0 \leq i < 16 : z_i \leftarrow x_i^0 + x_i^R \mod 2^{32} \ . \tag{11}$$

Note that the Salsa20 specification [5] defines both a `columnround` and a `rowround` function. In this paper, we include a matrix transposition as part of every round function. This simplifies the analysis: because of our definitions, every round of Salsa20 is identical.

5 Finding Differential Trails using SAT Solvers (ePrint 2013/328)

The main insight of ePrint 2013/328 [26] is that the differential trail of a certain weight can be described as the solution of a Boolean formula for any given ARX cipher. These Boolean formulas can then be converted into conjunctive normal form (CNF), and solved using a Boolean Satisfiability Problem (SAT) solver.

We are interested not only in the solutions that are given by the SAT solver, but even more in the cases where the SAT solver does not return a solution:

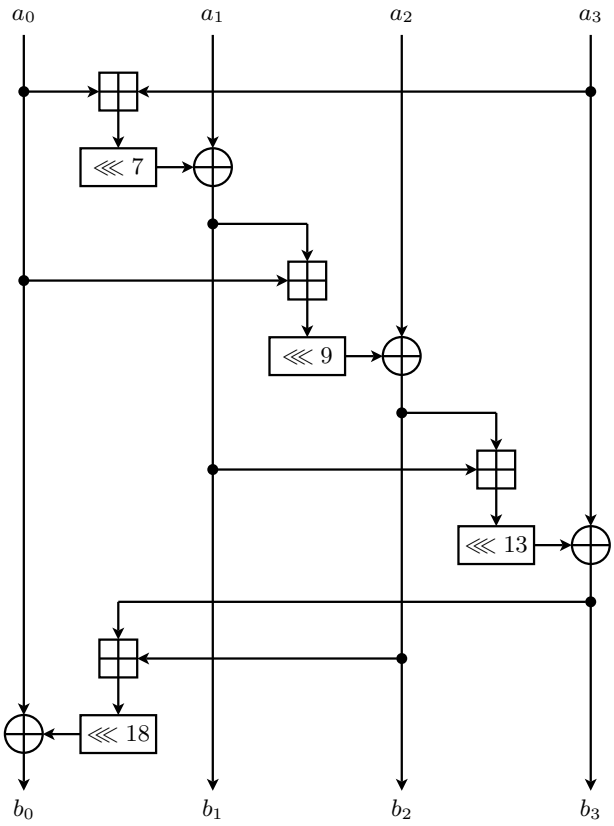

Fig. 2. The Salsa20 `quarterround` function is defined as: $(b_0, b_1, b_2, b_3) \leftarrow$ `quarterround`(a_0, a_1, a_2, a_3)

a *complete* SAT solver will only output *unsatisfiable* when no solution exists, thereby proving that no trail exists with the given constraints.

To make the modeling of the problem easier, we will use a Satisfiability Modulo Theories (SMT) solver: this allows us to write expressions involving n-bit words. Often an SMT solver is a frontend to a SAT solver.

- For every pair of n-bit input words (x, x') of the cipher, we use one n-bit word Δx in the SMT solver to represent the XOR differences between the corresponding inputs $\Delta x = x \oplus x'$.
- Additional n-bit variables may be needed to represent the XOR differences of the outputs of the addition, XOR and rotate operations. These are introduced when required.
- For every XOR and every bit rotation in the ARX cipher, we apply the same XOR and bit rotation to the XOR differences. These hold with probability one, and are therefore not included in the weight calculation.

– For every addition modulo 2^n in the ARX cipher, we use (1) and (2) to ensure that the input and output differences correspond to valid differentials of the addition modulo 2^n. These equations ensure that either all differentials are valid, or SAT solver will output *unsatisfiable*.
– Additionally for every addition modulo 2^n of the ARX cipher, we include (4) to calculate the weight of the differential. This formula only applies to valid differentials, but this is ensured by the previous equations.
– The weights of all these differentials are summed together. We specify that the corresponding sum is at most W, which is the maximum weight of the differentials that are considered by our search program.
– We specify that at least one XOR input difference is non-zero. Otherwise, we would find the following trivial differential: if there is no difference in the inputs, there is no difference in the outputs with probability one.

The execution time of the SAT solver is difficult to predict. Experimentally, we found that the SAT solver only terminates within a reasonable amount of time for a small number of rounds. But if the number of rounds is too small, we obtain low-weight trails that don't help much to bound the weight of trails for the full-round cipher. For Salsa20, we focused on three-round trails, as these were the longest trails that we could search within a reasonable amount of time.

6 Differential Trails for Three-Round Salsa20

Salsa20 has 16 input variables: x_0^0, x_1^0,..., x_{15}^0. For each of these, we introduce a 32-bit variable to represent the XOR difference in the SMT solver. We can then straightforwardly apply the framework of Sect. 5 to find differentials for any number of rounds of Salsa20. However, there is one additional issue that should be taken into account.

For any number of rounds of Salsa20, there exist differential trails that have probability one. In particular, if $\forall i : 0 \leq i < 16 : x_i^r[31]$ are flipped, then $\forall i : 0 \leq i < 16 : x_i^{r+1}[31]$ will be flipped as well with probability one. This property was noted by several cryptographers, including Robshaw [4], Wagner [34] and Hernandez-Castro et al. [11].

As already pointed out by Bernstein [4], the use of the four 32-bit constants (c_0, c_1, c_2, c_3) ensures that these probability-one trails will never occur as an input to the Salsa20 round function. To avoid finding these probability-one trails in our search program, we arbitrarily specify that $\Delta x_0^0[31] = 0$. This also halves the number of trails found by our program: for every trail found, there exists another trail where the differences in every MSB are flipped.

6.1 Trails and Differentials

The results found for three-round Salsa20 in ePrint 2013/328 [26] are as follows. They can be reproduced using the software toolkit that is available online [25].

The tool returns *unsatisfiable* for trails with a weight below 18, proving that no solutions exist that satisfy the Boolean formula. The tool is used to enumerate

all 6,761,988 three-round Salsa20 trails with weights up to 26. For each of the trails found, no combination of input and output differences occurs more than once. Therefore, they correspond to 6,761,988 three-round Salsa20 differentials.

A straightforward modification of the tool quickly find all trails that belong to a certain differential. Out of all 6,761,988 three-round Salsa20 differentials, only 2,604 differentials contain more than one trail. The results are shown in Table 1: for all the differentials that we have found, only a very small fraction contains more than one trail, and this increases the weight of the differential only slightly.

Still, this represents a problem if we want to bound the differential probability. For some of the trails that we found of weight 24 and higher, the trail probability is not the same as the differential probability.

Table 1. For three rounds of Salsa20, we give the number of trails with weights up to 26, which we find in this case is the same as the number of differentials. Some of these differentials consist of more than one trail: we show for how many differentials this holds and estimate the weight of the best differential in every group.

Weight of Trail	Corresponding # Differentials	# Differentials with Multiple Trails	Weight of Best Differential
18	8	0	18
19	112	0	19
20	872	0	20
21	5,080	0	21
22	24,696	0	22
23	105,128	0	23
24	404,272	16	23.91
25	1,435,784	264	24.68
26	4,786,036	2324	25.68
Total	6,761,988	2604	

We give two examples of differentials with more than one trail: Differential A and Differential B. From now on, let us use the following notation to denote the XOR input and output differences of a differential:

$$\begin{bmatrix} x_0^0 & x_1^0 & x_2^0 & x_3^0 \\ x_4^0 & x_5^0 & x_6^0 & x_7^0 \\ x_8^0 & x_9^0 & x_{10}^0 & x_{11}^0 \\ x_{12}^0 & x_{13}^0 & x_{14}^0 & x_{15}^0 \end{bmatrix} \rightarrow \begin{bmatrix} x_0^3 & x_1^3 & x_2^3 & x_3^3 \\ x_4^3 & x_5^3 & x_6^3 & x_7^3 \\ x_8^3 & x_9^3 & x_{10}^3 & x_{11}^3 \\ x_{12}^3 & x_{13}^3 & x_{14}^3 & x_{15}^3 \end{bmatrix} \tag{12}$$

where all XOR differences will be denoted as hexadecimal values.

First, we consider Differential A:

$$\begin{bmatrix} \texttt{00000420} & \texttt{00001080} & \texttt{00200000} & \texttt{01000000} \\ \texttt{84021000} & \texttt{00021000} & \texttt{02000000} & \texttt{04000000} \\ \texttt{00084000} & \texttt{01004000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{01080000} & \texttt{88200000} & \texttt{00001001} & \texttt{00020000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00001000} & \texttt{40020000} & \texttt{00000000} & \texttt{80000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \end{bmatrix}.$$

In Fig. 3, we show the two trails that correspond to Differential A: a trail of weight 25 (left), as well as one of weight 27 (right), resulting in a differential weight of $-\log_2(2^{-25}+2^{-27}) = 24.67$. No other trails exist for Differential A.

In the next section, we will explain that the weight of the trail of Fig. 3 (right) does not correspond to the sum of the weights of every addition of the ARX cipher. We will perform a more accurate estimate to obtain a trail weight of 26, and therefore the differential has an estimated weight of $-\log_2(2^{-25}+2^{-26}) = 24.42$

Also note that the best found trail for Differential A cannot be obtained by a greedy search strategy: for the trail of Fig. 3 (left), the output difference after every addition is not always the one with the highest probability.

We then consider Differential B:

$$\begin{bmatrix} \texttt{00000010 00000840 00040100 80080000} \\ \texttt{00000800 21000000 84000000 00000100} \\ \texttt{00002000 80042010 80000010 00001000} \\ \texttt{00000000 00000042 00000802 00000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000 00000000 00000000 00000000} \\ \texttt{00000000 00000000 00000000 00000000} \\ \texttt{00040000 80000000 00020010 00000000} \\ \texttt{00000000 00000000 00000000 00000000} \end{bmatrix} .$$

Differential B is composed of trails with estimated weights of 24, 28 and 32, as well as trails of a higher weight. In Fig. 4, we only show the trail with estimated weights of 24 and 28. We again find that the estimated weight of the trails is not equal to the sum of the estimated weights of every component. A more accurate theoretical estimate, which will be performed in the next section, will assign to these three trails the estimated weights of 24, 27 and 30 respectively.

6.2 Probability of the Trails

As already mentioned in the previous section, the commonly made assumption that the probability of a differential trail is equal to the multiplication of the probabilities of each operation is not always correct.

To understand the particular problem that we found, we use the concept of signed differences $\Delta^{\pm}x$. These split up the XOR differences into three possible cases:

- $x[i] = x'[i]$, which is denoted as $\Delta^{\pm}x[i] = 0$,
- $x[i] = 0, x'[i] = 1$, which is denoted as $\Delta^{\pm}x[i] = +1$,
- $x[i] = 1, x'[i] = 0$, which is denoted as $\Delta^{\pm}x[i] = -1$.

Note that a signed difference $\Delta^{\pm}x$ corresponds to exactly one XOR difference Δx:

$$\Delta x = \bigoplus_{i=0}^{n-1} |\Delta^{\pm}x[i]| \cdot 2^i \ , \tag{13}$$

as well as to exactly one additive difference $\Delta^{+}x$:

$$\Delta^{+}x = \sum_{i=0}^{n-1} \Delta^{\pm}x[i] \cdot 2^i \mod 2^n \ . \tag{14}$$

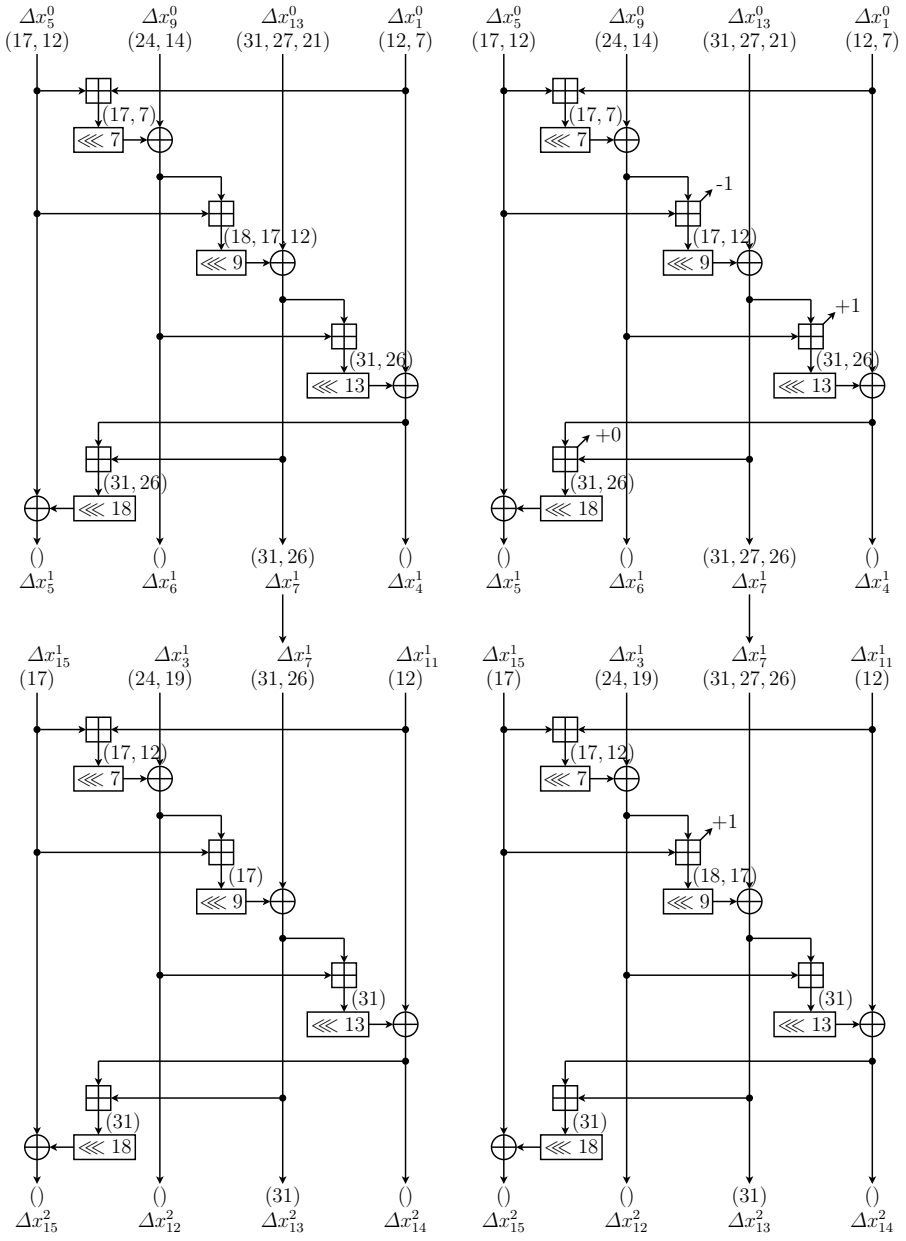

Fig. 3. Differential A (defined in Sect. 6.1) is composed of exactly two trails, one of weight 25 (left), and another of weight 26 (right). For the trail on the right, we use '+1' and '-1' to denote the increase or decrease of weight, compared to the trail on the left. In Sect. 6.2, we show that not all operations are independent for the trail on the right: this explains why one addition has '+0' instead of '+1'. Note that only part of the trails are shown.

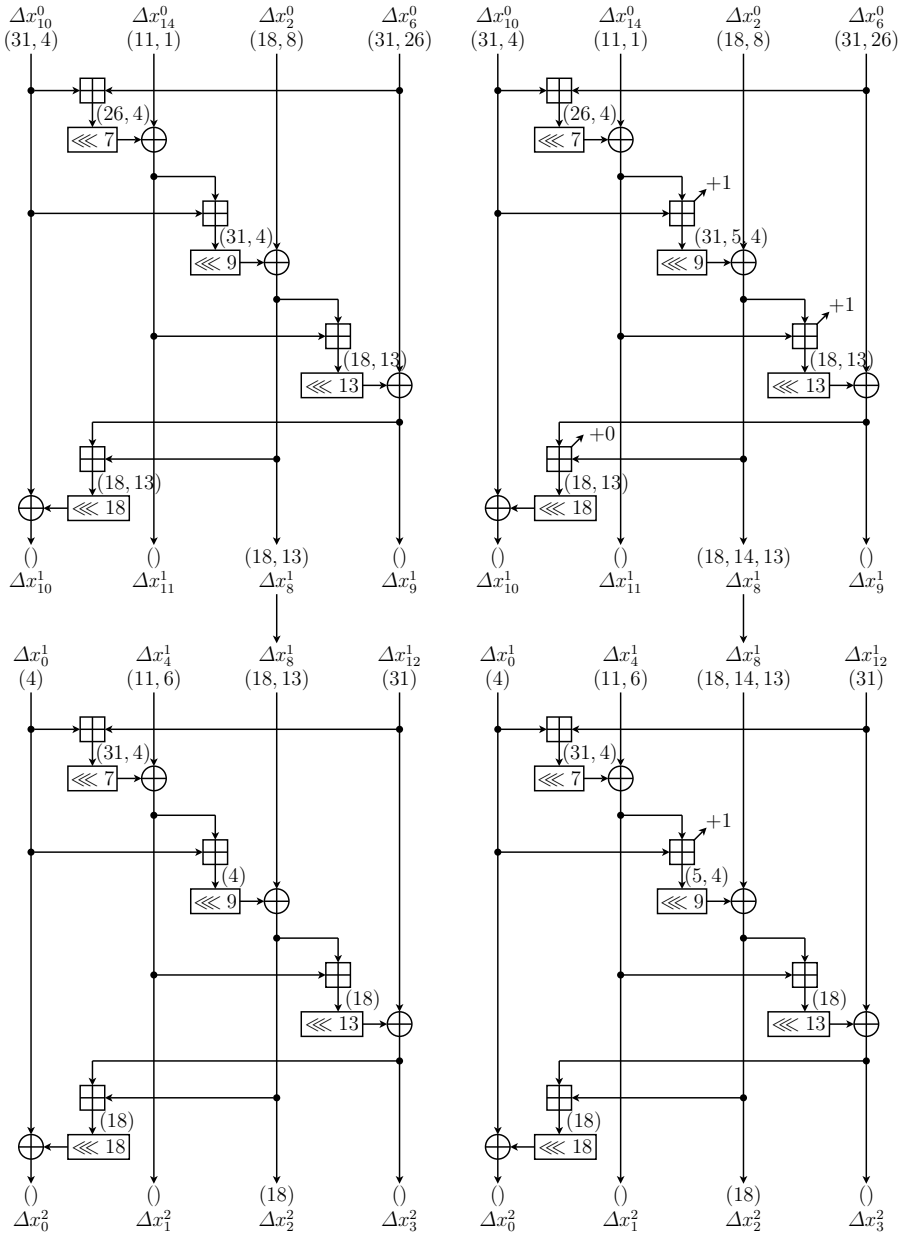

Fig. 4. Differential B (defined in Sect. 6.1) is composed of several trails. We show the two with the lowest weight: weight 24 (left), and weight 27 (right). For the trail on the right, we use '+1' and '-1' to denote the increase or decrease of estimated weight, compared to the trail on the left. In Sect. 6.2, we show that not all operations are independent for the trail on the right: this explains why one addition has '+0' instead of '+1'. Note that only part of the trails are shown.

Let us now revisit Differential A. In 3 (right), we find an addition with input differences $\Delta x_7^1 = 2^{31} \oplus 2^{27} \oplus 2^{26}$ and $\Delta x_6^1 = 0$. Let us denote the corresponding output difference by $\Delta d = 2^{31} \oplus 2^{26}$.

Every XOR difference corresponds to a set of signed differences. Not all of these are valid for the addition operation. For example, $\Delta^{\pm} x_7^1[27] = +1$, $\Delta^{\pm} x_4^1[26] = -1$ and $\Delta^{\pm} d[26] = +1$ results in a valid assignment, because $2^{31} + 2^{27} - 2^{26} = 2^{31} + 2^{26} \mod 2^{32}$. However, no valid assignment exists if $\Delta^{\pm} x_7^1[27] = \Delta^{\pm} x_7^1[26]$.

Then, we observe that Δx_7^1 is reused in another addition which also imposes the condition $\Delta^{\pm} x_7^1[27] = \Delta^{\pm} x_7^1[26]$ to obtain a valid output difference. Because this condition is already satisfied, the differential probability of this addition increases from 2^{-2} to 2^{-1}.

The same effect occurs with Differential B, as can be seen from Fig. 4. Here, we see that the trail weight of Fig. 4 (right) can be more accurately estimated as 27 instead of 28.

The effect is also not limited to cases where a differential consists of several trails. Let us consider Differential C:

$$\begin{bmatrix} \texttt{00000000 00000000 00000002 10000002} \\ \texttt{00000000 80000000 00000040 00000108} \\ \texttt{00000000 00000040 03000000 00200000} \\ \texttt{00000000 00000100 80002000 04000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000 00000000 00000000 00000000} \\ \texttt{10280000 840040a2 00000040 00008100} \\ \texttt{00000000 00000000 00000000 00000000} \\ \texttt{00000000 00000000 00000000 00000000} \end{bmatrix} .$$

and Differential D:

$$\begin{bmatrix} \texttt{00000000 00000000 00000002 10000002} \\ \texttt{00000000 80000000 00000040 00000108} \\ \texttt{00000000 00000040 03000000 00200000} \\ \texttt{00000000 00000100 80002002 04000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000 00000000 00000000 00000000} \\ \texttt{10280000 840040a2 00000040 00008100} \\ \texttt{00000000 00000000 00000000 00000000} \\ \texttt{00000000 00000000 00000000 00000000} \end{bmatrix} .$$

Differentials C and D each contain exactly one trail, which are shown in Fig. 5 (left) and Fig. 5 (right) respectively. As shown by Fig. 5 (left), Differential C has a probability of 20 instead of 21 when an analysis of signed differences is taken into account. In Fig. 5 (right), the trail with estimated weight 22 turns out to be impossible due to a contradiction.

6.3 Experimental Verification of the Probabilities

In this section, we evaluate the probabilities of three-round Salsa20 trails experimentally. We evaluate the probability of Differentials A and B of Sect. 6.1, and Differentials C and D of Sect. 6.2.

We also selected nine differentials at random for three rounds of Salsa20, one for every weight from 18 up to 26. These trails are chosen at random with the following requirements: $\Delta x_0^0[31] = 0$, there is only one trail per differential, and no extra conditions appear due to modular differences. Note that for every differential where $\Delta x_0^0[31] = 0$, we can obtain another differential where $\Delta x_0^0[31] = 1$ with the same probability.

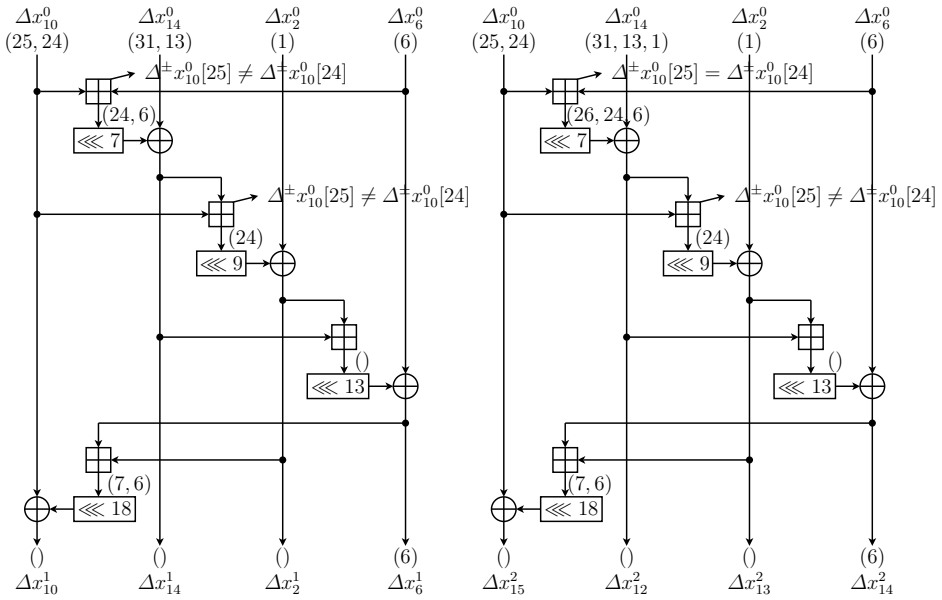

Fig. 5. Differential C (defined in Sect. 6.2) consists of exactly one trail (left), with a weight of 20 instead of weight 21 due to the condition $\Delta^{\pm} x_{10}^0[25] \neq \Delta^{\pm} x_{10}^0[24]$. Differential D (see Sect. 6.2) also consists of one trail (right). Although a first estimate showed that this trail had a weight of 21, we actually find that the trail is impossible due to a contradiction for $\Delta^{\pm} x_{10}^0[25]$ and $\Delta^{\pm} x_{10}^0[24]$. Note that only part of the trail are shown.

We selected the sample sizes such that the difference between the experimental weight and the theoretical weight will be at most 0.015 for 99% of the samples. The results of our calculations are given in Table 2. We find that all experimental weights pass a two-tailed binomial test with a significance level of 1%, which provides evidence that the experimental probability corresponds to the theoretical probability.

These results were computed after taking into account trail clustering and the improved estimate using signed differences. Without those corrections, the experiments give examples where the experimental probability contradicts the theoretical probability using the simple technique of Sect. 5.

7 Technical Discussion

Linearization is a common technique to find low-weight trails for ARX ciphers. Using this technique, every addition is replaced by XOR, which results in a linear code. Standard techniques from coding theory can then be used to find low-weight codewords [10, 28]. Linearization is a very powerful technique, and can find three-round Salsa20 trails of weight 18 in only a few seconds. All three-

Table 2. Experimental Probabilities of Differentials for Three Rounds of Salsa20. The differentials are specified in ePrint 2013/328 [26].

Name	Theoretical Weight	Sample Size	Expected Value	Experimental Value	Difference	p-value
Differential 18	18	2^{34}	65 536	65 800	+264	0.3033
Differential 19	19	2^{35}	65 536	65 209	-327	0.2022
Differential 20	20	2^{36}	65 536	65 496	-40	0.8774
Differential 21	21	2^{37}	65 536	65 664	+128	0.6185
Differential 22	22	2^{38}	65 536	65 667	+131	0.6102
Differential 23	23	2^{39}	65 536	65 119	-417	0.1037
Differential 24	24	2^{40}	65 536	65 725	+189	0.4615
Differential 25	25	2^{41}	65 536	65 113	-423	0.0989
Differential 26	26	2^{42}	65 536	65 848	+312	0.2237
Differential A	24.42	2^{41}	98 304	98 264	-40	0.8997
Differential B	23.81	2^{40}	74 896	75 227	+331	0.2272
Differential C	20	2^{36}	65 536	65 453	-83	0.7473
Differential D	∞	2^{42}	0	0	0	1

round trails of weight 18 given in Table 1, can be found by linearization. However, to find all linearized trails of weight 18 requires more than 19 days of computation on a 2.93 GHz Intel Xeon X7350 processor using MAGMAs MinimumWords function. The tool finds the same trails in only half an hour. Furthermore, linearization cannot find the non-linear three-round trails that exist for weight 19 and higher. This is clearly an advantage of the tool over previous techniques.

But the tool also has some non-obvious shortcomings, which were not captured by previous techniques. For ARX ciphers, Leurent proposed conditions on two and on three adjacent bits of one word in [20]. Although these conditions capture the conditions that encounter in Sect. 6.1 and Sect. 6.2, they do not detect the necessary conditions that appear in Salsa20 trails. For example, consider the 32-bit addition operation $a + b = c$, where $\Delta a = 2^3 \oplus 2^0$, $\Delta b = 0$ and $\Delta c = 2^2 \oplus 2^1 \oplus 2^0$. This differential is only valid if $\Delta^{\pm} a[3] \neq \Delta^{\pm} a[0]$. Leurent's ARXtools does not detect this condition, because it only considers three adjacent bits. Instead, it is necessary to convert one XOR difference into a signed difference, to allow ARXtools to detect this condition.

As we mentioned earlier, and as can be seen from Fig. 2, the output of an addition in Salsa20 is never used directly as the input of another addition. Instead, the output of the addition of two variables is, after rotation, XORed with a third variable. This, combined with the fact that the trails are very sparse, seems to ensure that the output values of every operation are uniformly distributed. If so, this would allow us add the differential weights of every operation, in order to obtain the differential weight of a differential for the entire cipher. Although most experiments give support that this assumption is correct, we found that the assumption can fail in unexpected ways.

In the first version of the paper, ePrint 2013/328 searched for all three-round trails below a certain weight, found that they cannot be connected to each other, and used this to bound the differential weight of a 15-round trail, leaving a few rounds of security margin for the 20-round Salsa20. However, the updated ePrint paper pokes some holes in the underlying assumptions, thereby invalidating the proof of resistance against differential cryptanalysis.

For completeness, we should point out this paper describes differential cryptanalysis only in its most basic form, and does not consider more advanced forms of differential cryptanalysis or other cryptanalysis techniques.

8 Philosophical Discussion

Given the novelty of organizing a "Conference for Failed Approaches and Insightful Losses (CFAIL)," it might not immediately be clear what the goal of such a conference would be. Perhaps CFAIL might be seen as a venue for unfinished work. However, the cryptographic community already offers ample opportunities for the early dissemination of work in progress, and it seems unlikely that such results would be "sitting in a drawer somewhere because it never quite panned out," as stated in the call for papers. Furthermore, CFAIL doesn't seem to be aiming for goofy research or humorous articles. Our understanding of the call for papers, is that it aims for papers that are:

- *Insightful*: Even if the problem is not solved, the approach should lead to new insights.
- *Well-written*: A breakthrough result might be spectacular enough to overlook bad writing, but the value in a failed approach seems to be mostly in the write-up. If the goal is to gain new insights, it is crucial that the article is well-written.
- *Inspiring*: If the approach failed, why tell the story at all? It seems important to inspire. Perhaps the cryptographic community is at risk of not focusing enough on long-standing open problems, fearing that papers will appear weaker if they point out all the shortcomings.

When building a theoretical model to calculate differential probabilities, it might seem naive in hindsight to assume that none of the assumptions would fail in practice. However, the paper started out to challenge the opposite point of view: it seemed that some cryptanalysts rely too much on empirical results, which can give a false feeling of confidence.

There is inherently a limitation to the number of experiments that we can perform, and statistical anomalies are too often brushed away. Perhaps cryptanalysts say too quickly that they "got lucky" or "were unlucky" after running their experiments, whereas perhaps more accurate theoretical modeling would have explained the discrepancies.

But even if the paper was aiming too high, the SAT-solver-based technique that it introduced to find differential trails, has gotten a life of its own. It is now described as one of the two "standard tools" to search for differential trails [23].

Interesting to note, is that the technique was already used in the design of Simon and Speck according to a recent report by the National Security Agency (NSA) [2]. Given that Simon and Speck were designed in 2012 or earlier [32], this would mean that the technique was already known to the NSA before it was independently rediscovered in ePrint 2013/328.

9 Conclusion

In an attempt to prove bounds for differential trails of ARX constructions, a new SAT-solver-based technique was developed to find differential trails in ePrint 2013/328. When applied to Salsa20, the technique didn't find any better trails than could be found through simple linearization, but proved the non-existence of some trails, in the hope of proving bounds on differentials.

Experiments showed that the assumptions needed to prove these bounds (claimed in the initial version on ePrint), fail to hold in practice (as explained in the updated version on ePrint). It may be that proving bounds is too ambitious, at least given the current state-of-the-art. But the technique used in the paper became of independent interest, and has been used to find better and more accurate trails for the design and analysis of other ciphers.

Acknowledgments. Certain algorithms and commercial products are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the algorithms or products identified are necessarily the best available for the purpose.

References

1. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg [31], pp. 470–488. https://doi.org/10.1007/978-3-540-71039-4_30
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: Notes on the design and analysis of SIMON and SPECK. Cryptology ePrint Archive, Report 2017/560 (2017), `https://eprint.iacr.org/2017/560`
3. Bernstein, D.J.: Salsa20/8 and Salsa20/12. `http://cr.yp.to/snuffle/812.pdf` (February 2006)
4. Bernstein, D.J.: Response to "On the Salsa20 core function". `http://cr.yp.to/snuffle/reoncore-20080224.pdf` (February 2008)
5. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986, pp. 84–97. Springer (2008). https://doi.org/10.1007/978-3-540-68351-3_8
6. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1

7. den Boer, B., Bosselaers, A.: Collisions for the Compressin Function of MD5. In: Helleseth [16], pp. 293–304. https://doi.org/10.1007/3-540-48285-7_26
8. Brassard, G. (ed.): Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, Lecture Notes in Computer Science, vol. 435. Springer (1990). https://doi.org/10.1007/0-387-34805-0
9. Browning, K.A., Dillon, J.F., McQuistan, M.T., Wolfe, A.J.: An APN permutation in dimension six. Proceedings of the 9th International Conference on Finite Fields and their Applications, Dublin, July 13-17, 2009, **518**, 33–42 (2010)
10. Canteaut, A., Chabaud, F.: A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Trans. Information Theory **44**(1), 367–378 (1998). https://doi.org/10.1109/18.651067
11. Castro, J.C.H., Estévez-Tapiador, J.M., Quisquater, J.: On the Salsa20 Core Function. In: Nyberg [31], pp. 462–469. https://doi.org/10.1007/978-3-540-71039-4_29
12. Daemen, J., Lamberger, M., Pramstaller, N., Rijmen, V., Vercauteren, F.: Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. Computing **85**(1-2), 85–104 (2009). https://doi.org/10.1007/s00607-009-0034-y
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). https://doi.org/10.1007/978-3-662-04722-4
14. Damgård, I.: A Design Principle for Hash Functions. In: Brassard [8], pp. 416–427. https://doi.org/10.1007/0-387-34805-0_39
15. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Submission to the NIST SHA-3 Competition (Round 3) (2010), `http://www.skein-hash.info/sites/default/files/skein1.3.pdf`
16. Helleseth, T. (ed.): Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings, Lecture Notes in Computer Science, vol. 765. Springer (1994). https://doi.org/10.1007/3-540-48285-7
17. Hong, D., Lee, J., Kim, D., Kwon, D., Ryu, K.H., Lee, D.: LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In: Kim, Y., Lee, H., Perrig, A. (eds.) Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8267, pp. 3–27. Springer (2013). https://doi.org/10.1007/978-3-319-05149-9_1
18. Hong, S., Hong, D., Ko, Y., Chang, D., Lee, W., Lee, S.: Differential Cryptanalysis of TEA and XTEA. In: Lim, J.I., Lee, D.H. (eds.) Information Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 2971, pp. 402–417. Springer (2003). https://doi.org/10.1007/978-3-540-24691-6_30
19. Kelsey, J., Schneier, B., Wagner, D.A.: Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 237–251. Springer (1996). https://doi.org/10.1007/3-540-68697-5_19

20. Leurent, G.: Analysis of Differential Attacks in ARX Constructions. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 226–243. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_15
21. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer (2001). https://doi.org/10.1007/3-540-45473-X_28
22. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: Rueppel, R.A. (ed.) Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings. Lecture Notes in Computer Science, vol. 658, pp. 81–91. Springer (1992). https://doi.org/10.1007/3-540-47555-9_7
23. Mella, S., Daemen, J., Assche, G.V.: New techniques for trail bounds and application to differential trails in Keccak. IACR Trans. Symmetric Cryptol. **2017**(1), 329–357 (2017). https://doi.org/10.13154/tosc.v2017.i1.329-357
24. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [8], pp. 428–446. https://doi.org/10.1007/0-387-34805-0_40
25. Mouha, N.: Toolkit for Bounding Characteristics using SAT/SMT Solvers. `https://mouha.be/tools/` (June 2013)
26. Mouha, N., Preneel, B.: Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328 (2013), `https://eprint.iacr.org/2013/328`
27. Mouha, N., Velichkov, V., De Cannière, C., Preneel, B.: The Differential Analysis of S-Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6544, pp. 36–56. Springer (2010). https://doi.org/10.1007/978-3-642-19574-7_3
28. Nad, T.: The CodingTool Library. In: Workshop on Tools for Cryptanalysis. pp. 129–130 (2010)
29. National Institute of Standards and Technology: Announcing the ADVANCED ENCRYPTION STANDARD (AES). NIST Federal Information Processing Standards Publication 197 (November 2001). https://doi.org/10.6028/NIST.FIPS.197
30. Nyberg, K.: Differentially Uniform Mappings for Cryptography. In: Helleseth [16], pp. 55–64. https://doi.org/10.1007/3-540-48285-7_6
31. Nyberg, K. (ed.): Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers, Lecture Notes in Computer Science, vol. 5086. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4
32. Saarinen, M.J.O., Engels, D.: A Do-It-All-Cipher for RFID: Design Requirements (Extended Abstract). Cryptology ePrint Archive, Report 2012/317 (2012), `https://eprint.iacr.org/2012/317`
33. Steil, M.: 17 Mistakes Microsoft Made in the Xbox Security System. 22nd Chaos Communication Congress (December 2005), `http://events.ccc.de/congress/2005/fahrplan/events/559.en.html`

34. Wagner, D.: Re-rolled Salsa20 function. http://groups.google.com/group/sci.crypt/msg/0692e3aaf78687a3 (September 2005)
35. Wheeler, D.J., Needham, R.M.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. Lecture Notes in Computer Science, vol. 1008, pp. 363–366. Springer (1994). https://doi.org/10.1007/3-540-60590-8_29