Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20*

Nicky Mouha[1,2] and Bart Preneel[1,2]

[1] Department of Electrical Engineering, ESAT/COSIC, KU Leuven, Belgium.
[2] iMinds, Belgium.
Nicky.Mouha@esat.kuleuven.be

Abstract. An increasing number of cryptographic primitives are built using the ARX operations: addition modulo 2^n, bit rotation and XOR. Because of their very fast performance in software, ARX ciphers are becoming increasingly common. However, there is currently no rigorous understanding of the security of ARX ciphers against one of the most common attacks in symmetric-key cryptography: differential cryptanalysis. In this paper, we introduce a tool to search for optimal differential characteristics for ARX ciphers. Our technique is very easy to use, as it only involves writing out simple equations for every addition, rotation and XOR operation in the cipher, and applying an off-the-shelf SAT solver. As is commonly done for ARX ciphers, our analysis assumes that the probability of a characteristic can be computed by multiplying the probabilities of each operation, and that the probability of the best characteristic is a good estimate for the probability of the corresponding differential. Using extensive experiments for Salsa20, we find that these assumptions are not always valid. To overcome these issues, we propose a method to accurately estimate the probability of ARX differentials.

Keywords: Differential cryptanalysis, ARX, Evaluation Tool, SAT solver, Salsa20.

1 Introduction

ARX ciphers are composed of only three operations: additions modulo 2^n, bit rotations and XORs. We use n to denote the word size, typically $n = 32$ or $n = 64$. Recently, many ciphers have appeared that are based on ARX. Examples are the SHA-3 finalist hash functions BLAKE [4] and Skein [19], the tweakable block cipher Threefish [19] and the stream cipher Salsa20 [8].

Although the term ARX was not introduced until 2009,[3] the concept of ARX ciphers is much older, and dates back to at least 1987, when the block

* This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and by the iMinds-MiX-MediaID project.

[3] Although Weinmann initially introduced the term AXR [41], he changed this into ARX soon afterwards. We use ARX in this paper, because this term has come into wide acceptance in cryptographic literature.

cipher FEAL was published [31]. In this paper, we will use the term "ARX" interchangeably with "pure ARX" [22, 23]. That is, we do not use ARX to refer to ciphers that also use other components as a source of non-linearity in GF(2), such as the bitwise Boolean functions used in MD5 and SHA-1.

The security of ARX ciphers is currently not well understood. To analyze the security of symmetric-key cryptographic primitives, two of the most powerful techniques are differential cryptanalysis [9] and linear cryptanalysis [25]. Security against these attacks is therefore typically a major design criterion for modern ciphers. For example, the block cipher AES used the wide-trail design strategy to provide provable resistance against both linear and differential cryptanalysis [16].

For ARX ciphers, there are currently no proven security bounds against differential cryptanalysis in existing literature. Because typical ARX ciphers are very fast on high-end processors, they are usually overdesigned. A large number of rounds is used, because the optimal trade-off between security and efficiency is not well understood. Designers of ARX ciphers have previously attempted to argue the security against differential cryptanalysis by using a variety techniques to search for high-probability differential characteristics, and explaining that high-probability differential characteristics could not be found for a sufficient number of rounds [4, 8, 19].

In this paper, we construct a tool to search for optimal differential characteristics of ARX ciphers. To use the tool, all that is required is to write simple equations for every addition, rotation and XOR of the ARX cipher. These equations are then input into STP [20], a program that converts these equations into a conjunctive normal form (CNF) formula. A SAT solver then either finds a satisfying assignment to the CNF formula, or outputs *unsatisfiable* when it can prove that no valid assignment exists.

Our technique assumes that the probability of the best characteristic is an accurate estimate of the probability of the corresponding differential. We also assume that the probability of the characteristic equals the product of the probabilities of each operation. These assumptions are commonly made, either implicitly or explicitly, when searching for characteristics for the ARX hash functions Skein [1,23,43] and BLAKE [3,36], as well as for MD5 [34,35] and SHA-1 [18,39].

Whereas most differential cryptanalysis attacks are based on theoretical calculations of probabilities, some attacks are based on experimentally derived probabilities. Notable examples include attacks by Crowley on Salsa20 [14], by Aumasson et al. on Salsa20, ChaCha and Rumba [2], and by Biryukov et al. on BLAKE [10]. However, such experimental estimations are computationally very intensive, and can therefore only be performed for a limited number of rounds. Thus it is important that we can easily and accurately estimate the probability of a characteristic.

We perform experiments to verify the probability of differentials, more specifically investigate if a differentials consist of multiple characteristics and if the differential probabilities of every operation can be multiplied. We find that our assumptions do not always hold. Therefore, we introduce techniques to accu-

Table 1. Notation.

Notation	Description
$x + y$	addition of x and y modulo 2^n (in text)
$x \boxplus y$	addition of x and y modulo 2^n (in figures)
$x \lll s$	rotation of x to the left by s positions
$x \ll s$	shift of x to the left by s positions
$x \oplus y$	bitwise exclusive OR (XOR) of x and y
$x \wedge y$	bitwise AND of x and y
$\neg x$	bitwise NOT of x
$:=$	is defined as
Δx	XOR difference of x and x': $\Delta x = x \oplus x'$
$\Delta^+ x$	additive difference of x and x': $\Delta x = x' - x'$
$\Delta^\pm x$	signed difference of x and x': $\Delta x[i] = x'[i] - x[i] \in -1, 0, 1$
$x[i]$	bit selection: bit at position i of word x, where $i = 0$ is the least significant bit
$(i, j, ...)$	XOR difference of $2^i \oplus 2^j \oplus ...$

rately estimate the probability of a differential, and then verify these techniques experimentally.

Outline. Notation is defined in Table 1. Section 2 gives an overview of related work. We define the Salsa20 stream cipher in Sect. 3. In Sect. 4, we provide a general framework to find differential characteristics for ARX ciphers. This framework is used to find three-round Salsa20 characteristics up to weight 26 in Sect. 5. We investigate the probability of these characteristics and their corresponding differentials, both theoretically and experimentally. We conclude in Sect. 6. The characteristics for three rounds of Salsa20 with the highest probability are given in App. A.1. In App. A.2, we give 9 randomly selected differentials, for which we experimentally verify probability in Sect. 5.5. Source code to find characteristics for three rounds of Salsa20 is given in App. A.3.

2 Related Work

Biham and Shamir introduced differential cryptanalysis in [9]. For block ciphers, it is used to analyze how input differences in the plaintext lead to output differences in the ciphertext. If this happens in a non-random way, this can be used to build a distinguisher or even a key-recovery attack for the block cipher. In the case of hash functions, differential cryptanalysis can be used to construct a collision attack, as was done for the commonly used hash functions MD5 [40] and SHA-1 [39]. For stream ciphers, differential cryptanalysis can be used in the context of a resynchronization attack [15].

To prove the security of ciphers against differential cryptanalysis, Lai et al. introduced the theory of Markov ciphers [21]. Their paper was the first to make

a distinction between a *differential* and a *differential characteristic*. A differential is a difference propagation from an input difference to an output difference. A differential characteristic defines not only the input and output differences, but also the internal differences after every round of the iterated cipher. The probability of a differential is equal to the sum of the probabilities of all differential characteristics that correspond to this differential. It is commonly assumed that the probability of the best differential can accurately be estimated by the probability of the best differential characteristic.

Daemen and Rijmen used this theory of Markov ciphers to prove the security of AES against differential cryptanalysis [16]. This was done by proving a lower bound for the probability of the best differential characteristic. For AES in the single-key setting, such a proof is very straightforward because of the wide trail design strategy.

For ARX ciphers, finding differential characteristics is much more difficult. It typically involves months of tedious manual calculations (as done by Wang et al. for several hash functions, including MD5 [40] and SHA-1 [39]), or the construction of a heuristic search program.

By their very nature, these heuristic search programs cannot guarantee that optimal characteristics are found. The search program either only considers a limited set of high-probability output differences for every operation [33, 34], or makes random guesses to construct characteristics [18]. Random guesses also appear in linearization, a technique where every ARX cipher is replaced by an XOR operation, after which techniques from coding theory are used to find low-weight codewords [11, 28].

In this paper, we propose a radically different approach. By use of a SAT solver, we do not limit ourselves to finding "good" characteristics, but instead we find optimal characteristics according to certain criteria.

Although our toolkit is not specific to one specific cipher, we chose to test our techniques on the ARX stream cipher Sala20. In the next section, we formally define Salsa20.

3 Description of Salsa20

Salsa20 is a stream cipher designed by Bernstein [8]. 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 [6]. For the sake of clarity, the 20-round Salsa20 is sometimes referred to as Salsa20/20.

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 [2], there are currently no known attacks on either Salsa20/12 or Salsa20/20.

Although eSTREAM portfolio includes Salsa20 with 12 rounds, Bernstein is more conservative and recommends choosing $R = 20$ for typical cryptographic applications [8].

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:

$$\begin{bmatrix} x_0^0 & x_1^0 & x_2^0 & x_3^0 \\ x_4^0 & x_5^0 & x_6^0 & x_7^0 \\ x_8^0 & x_9^0 & x_{10}^0 & x_{11}^0 \\ x_{12}^0 & x_{13}^0 & x_{14}^0 & x_{15}^0 \end{bmatrix} \leftarrow \begin{bmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{bmatrix} \quad . \tag{1}$$

A Salsa20 round consists of four parallel `quarterround` functions, defined in Fig. 1:

$$(y_0^r, y_4^r, y_8^r, y_{12}^r) \leftarrow \texttt{quarterround}(x_0^r, x_4^r, x_8^r, x_{12}^r) \ , \tag{2}$$
$$(y_5^r, y_9^r, y_{13}^r, y_1^r) \leftarrow \texttt{quarterround}(x_5^r, x_9^r, x_{13}^r, x_1^r) \ , \tag{3}$$
$$(y_{10}^r, y_{14}^r, y_2^r, y_6^r) \leftarrow \texttt{quarterround}(x_{10}^r, x_{14}^r, x_2^r, x_6^r) \ , \tag{4}$$
$$(y_{15}^r, y_3^r, y_7^r, y_{11}^r) \leftarrow \texttt{quarterround}(x_{15}^r, x_3^r, x_7^r, x_{11}^r) \ , \tag{5}$$

followed by a matrix transposition:

$$\forall i, j : 0 \leq i < 4, 0 \leq j < 4 : x_{4i+j}^{r+1} \leftarrow y_{4j+i}^r \ . \tag{6}$$

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

$$\forall i : 0 \leq i < 16 : z_i \leftarrow x_i^0 + x_i^R \mod 2^{32} \ . \tag{7}$$

Note that Salsa20 specification [8] 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.

4 A Framework to Find Differential Characteristics of ARX Ciphers

4.1 Calculating Differential Probabilities

Given a differential characteristic, we want to calculate the probability with which it holds. Unless explicitly mentioned otherwise, we always use XOR differences. We say that a differential is *valid* if it occurs with non-zero probability. The bit rotation and XOR operations are linear in GF(2). Therefore, for every input difference, there is only one valid output difference.

In [24], Lipmaa and Moriai study the differential properties of addition. Let $\text{xdp}^+(\alpha, \beta \rightarrow \gamma)$ be the XOR-differential probability of addition modulo 2^n, with

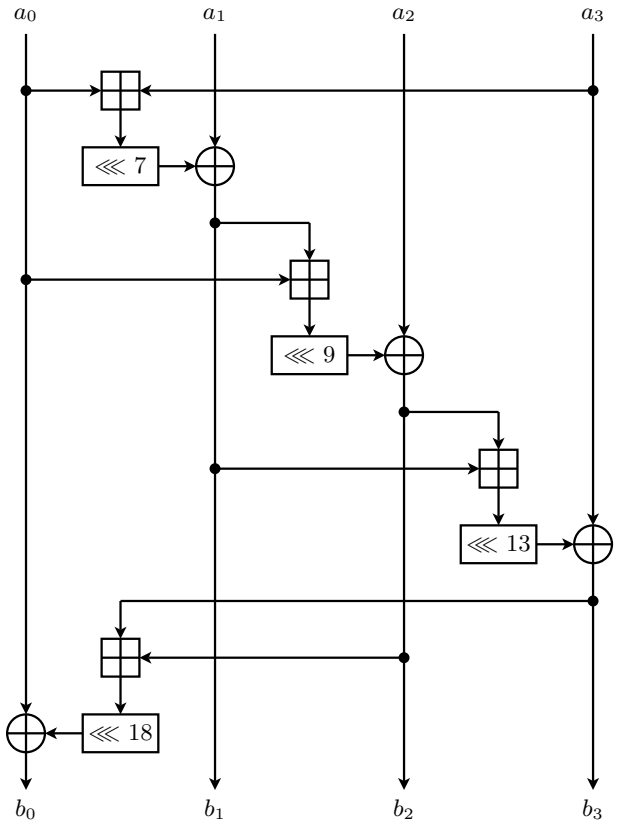

Fig. 1. The Salsa20 `quarterround` function is defined as: $(b_0, b_1, b_2, b_3) \leftarrow \texttt{quarterround}(a_0, a_1, a_2, a_3)$

input differences α and β and output difference γ. Lipmaa and Moriai prove that the differential $(\alpha, \beta \rightarrow \gamma)$ is valid if and only if:

$$\texttt{eq}(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0 \ , \tag{8}$$

where

$$\texttt{eq}(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z) \ . \tag{9}$$

For every valid differential $(\alpha, \beta \rightarrow \gamma)$, we define the *weight* $w(\alpha, \beta \rightarrow \gamma)$ of the differential as follows:

$$w(\alpha, \beta \rightarrow \gamma) := -\log_2(\mathrm{xdp}^+(\alpha, \beta \rightarrow \gamma)) \ . \tag{10}$$

The weight of a valid differential can then be calculated as:

$$w(\alpha, \beta \rightarrow \gamma) := h^*(\neg\texttt{eq}(\alpha, \beta, \gamma)) \ , \tag{11}$$

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

In our analysis, we assume that the probability of a valid differential characteristic is equal to the multiplication of the probabilities of each addition operation. Put differently, we calculate the weight of a valid differential characteristic as the sum of the weights of each addition operation. In Sect. 5.4, we will revisit this assumption by investigating cases where this assumption does not hold.

4.2 Searching for Differential Characteristics

In this paper, we will use a SAT solver to find differential characteristics up to a certain weight W. If a *complete* SAT solver returns *unsatisfiable*, this proves that no such differential characteristic exists. Initially, we choose W to be sufficiently small, so that the search space is limited and the SAT solver terminates within a reasonable time. We then gradually increase W and run the solver again, in order to search for characteristics of a higher weight.

The first problem that we encounter, is that the input of typical SAT solvers must be in conjunctive normal form (CNF). This corresponds to a product-of-sum with binary variables. The expressions that we will use, however, involve Boolean functions and additions on n-bit words, where $n = 32$ in the case of Salsa20. To overcome this problem, we will make use of STP [20]. STP converts equations using n-bit words into CNF, and then invokes a SAT solver. If a satisfying solution exists, STP converts the solution back into a solution for the original n-bit words.

To search for differential characteristics, we proceed with the following steps:

- For every pair of n-bit input words (x, x') of the cipher, we use one n-bit word Δx in STP 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 (8) and (9) 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 (11) 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 STP program will be generated by a small script, that processes the addition operation in a special way to generate the corresponding equations. Note that using this method, the input to our script corresponds almost exactly to the C source code of the ARX algorithm. This greatly simplifies the task of the cryptanalyst, and minimizes the possibility that a human error is made.

5 Application to Salsa20

5.1 Applying the Framework

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 STP. We can then straightforwardly apply the framework of Sect. 4 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 characteristics 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 [7], Wagner [37] and Hernandez-Castro et al. [12].

As already pointed out by Bernstein [7], the use of the four 32-bit constants (c_0, c_1, c_2, c_3) ensures that these probability-one characteristics will never occur as an input to the Salsa20 round function. To avoid finding these probability-one characteristics in our search program, we arbitrarily specify that $\Delta x_0^0[31] = 0$. This also halves the number of characteristics found by our program: for every found characteristic, there exists another characteristic where the differences in every MSB are flipped.

5.2 Characteristics for Three Rounds of Salsa20

In what follows, we will describe how to apply our toolkit to three rounds of Salsa20. The basic program can be found in App. A.3, and is part of a software toolkit that will be made available online.

Weight 17. We include equations for three rounds of Salsa20, and specify that we want to find all differential characteristics up to weight $W = 17$. We solve this problem using STP. The default SAT solver of STP, CryptoMiniSat2 [32], is used as a back end. The system that we will use for all our experiments, is a 3.4 GHz Intel Core i7-2600 processor. After 23 minutes and 38 seconds of computation, the SAT solver outputs *unsatisfiable*. Therefore, we find that three rounds of Salsa20, no differential characteristic exists with a weight of less than 18.

Weight 18. To find characteristics with weight 18, we proceed in a similar way, but now with $W = 18$ as a bound. We are interested in finding not just one, but all differential characteristics. Because this is not supported by STP, we tell STP to display the CNF formula and exit.

We input this CNF formula into a SAT solver. For every solution that we find, we generate an extra clause that blocks this solution, and then run the SAT solver again. This is repeated until the SAT solver returns *unsatisfiable*, which tells us that no more solutions exist.

If we use an *incremental* SAT solver, the SAT solver does not start again from scratch when we add a new clause, but instead retains previously learned information. CryptoMiniSat2 is an incremental SAT solver, and can be used to find all solutions using the blocking clauses method with the `--maxsolutions` flag. After 26 minutes and 22 seconds, we find four differential characteristics before CryptoMiniSat2 returns *unsatisfiable*.

These four characteristics satisfy $\Delta x_0^0[31] = 0$, as explained earlier. By flipping the MSB of every XOR word difference, the other four differential characteristics of weight 18 can be constructed. The differential characteristics are shown in App. A.1. Note that four characteristics identical except for a reordering of the input XOR differences. This is a result of the symmetry of the Salsa20 core function, as already pointed out in the original design document [5, 7].

Weight 26. For three rounds of Salsa20, the number of characteristics increases exponentially when the weight is increased. Adding blocking clauses for each of these solutions greatly increases the number of clauses that the SAT solver has to deal with. We must therefore use a different approach, otherwise the SAT solver will eventually run out of memory after several days of computation.

However, we have noticed that although a very large number of characteristics exist with weight up to 26, many of these characteristics share the same internal differences. We have constructed an STP program to find characteristics up to $W = 26$, with $\Delta x_0^0[31] = 0$. STP generates a CNF program, which we again solve with CryptoMiniSat2. However, for every characteristic that we find, we add a blocking clause that contains only Δx_0^2, Δx_1^2,..., Δx_{15}^2. After 18 hours, 32 minutes and 9 seconds, the SAT solver that there are 418 possible values for the input difference to the third round.

To enumerate all these characteristics is straightforward: using the tool of App. A.3, we construct 418 SAT programs that specify each of the 418 inputs differences to the third round, and again use CryptoMiniSat2 to find all solutions. We find a total of 6,761,988 characteristics. In the next section, we study these characteristics in detail.

5.3 Differentials and Characteristics

From Sect. 2, we recall that the probability of a differential is equal to the sum of the probabilities of all differential characteristics that correspond to this differential. Using our framework, it is easy to find all characteristics that correspond

to a given differential. When we specify the input and output differences, it takes CryptoMiniSat2 only a fraction of a second to find all characteristics.

For each of the 6,761,988 three-round Salsa20 characteristics of weight up to 26 found in the previous section, we find that no combination of input and output differences occurs more than once. As such, we obtain 6,761,988 three-round Salsa20 differentials. We find that out of these, only 2,604 differentials contain more than one characteristic.

Using the weights of all characteristics corresponding to a differential, we obtain an estimate for the weight of a differential. The results are shown in Table 2: for all the differentials that we have found, only a very small fraction contains more than one characteristic, and this increases the weight of the differential only slightly.

Table 2. For three rounds of Salsa20, we give the number of characteristics 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 characteristic: we show for how many differentials this holds and estimate the weight of the best differential in every group.

Weight of Characteristic	Corresponding # Differentials	# Differentials with Multiple Characteristics	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 characteristic: 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:

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

In Fig. 2, we show the two characteristics that correspond to Differential A: a characteristic 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 characteristics exist for Differential A.

In the next section, we will explain that the weight of the characteristic of Fig. 2 (right) does not correspond to the sum of the weights of every addition of the ARX cipher. We will calculate that this characteristic actually has a weight of 26, and therefore the differential has a weight of $-\log_2(2^{-25} + 2^{-26}) = 24.42$

Also note that the best found characteristic for Differential A cannot be obtained by a greedy search strategy: for the characteristic of Fig. 2 (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 characteristics with weights 24, 28 and 32, as well as characteristics of a higher weight. In Fig. 3, we only show the characteristics with estimated weights of 24 and 28. We again find that the weight of the characteristic is not equal to the sum of the weights of every component. A more detailed calculation, which will be performed in the next section, will show that these three characteristics have actual weights of 24, 27 and 30 respectively.

5.4 Probability of the Characteristics

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

To understand why, 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:

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

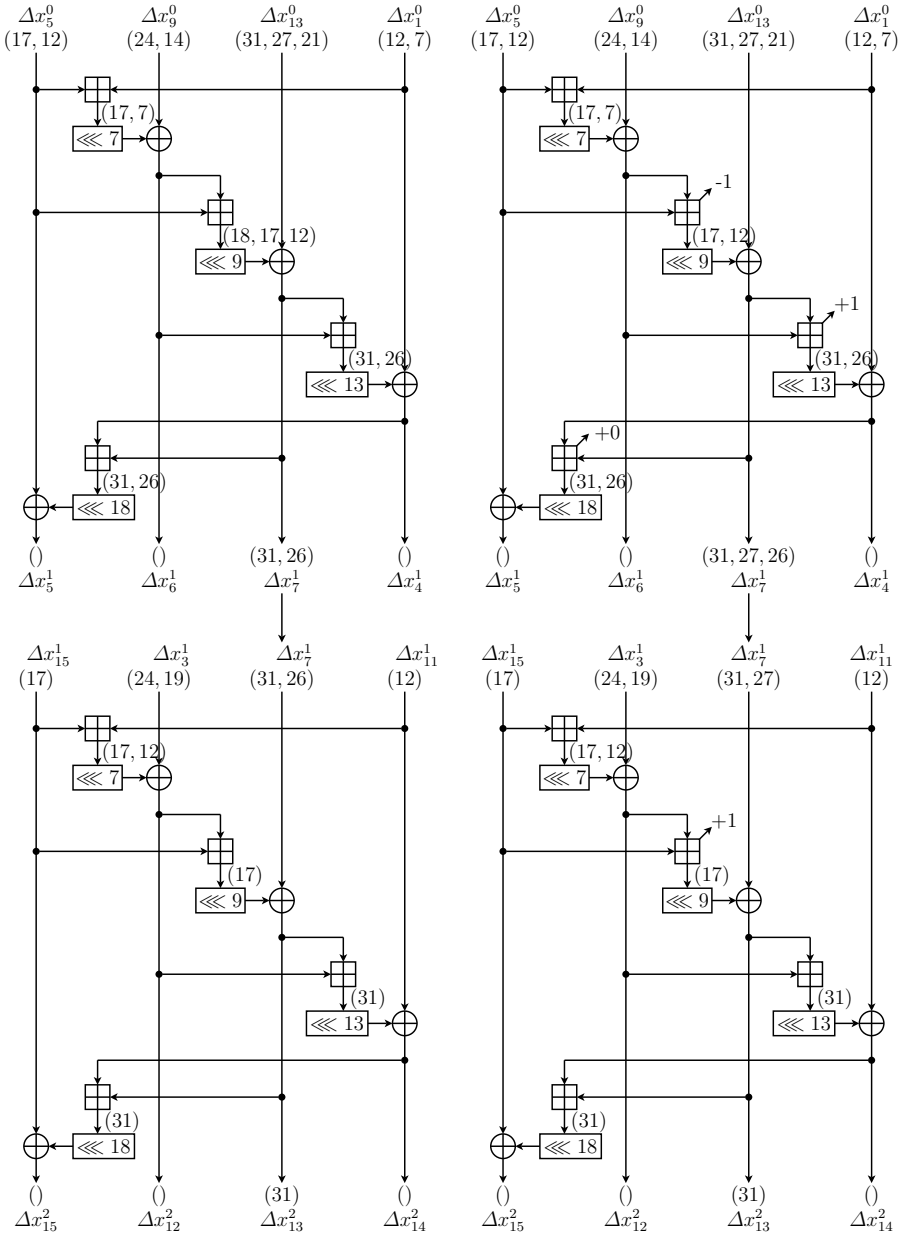

Fig. 2. Differential A (defined in Sect. 5.3) is composed of exactly two characteristics, one of weight 25 (left), and another of weight 26 (right). For the characteristic on the right, we use '+1' and '-1' to denote the increase or decrease of weight, compared to the characteristic on the left. In Sect. 5.4, we show that not all operations are independent for the characteristic on the right: this explains why one addition has '+0' instead of '+1'. Note that only part of the characteristics are shown.

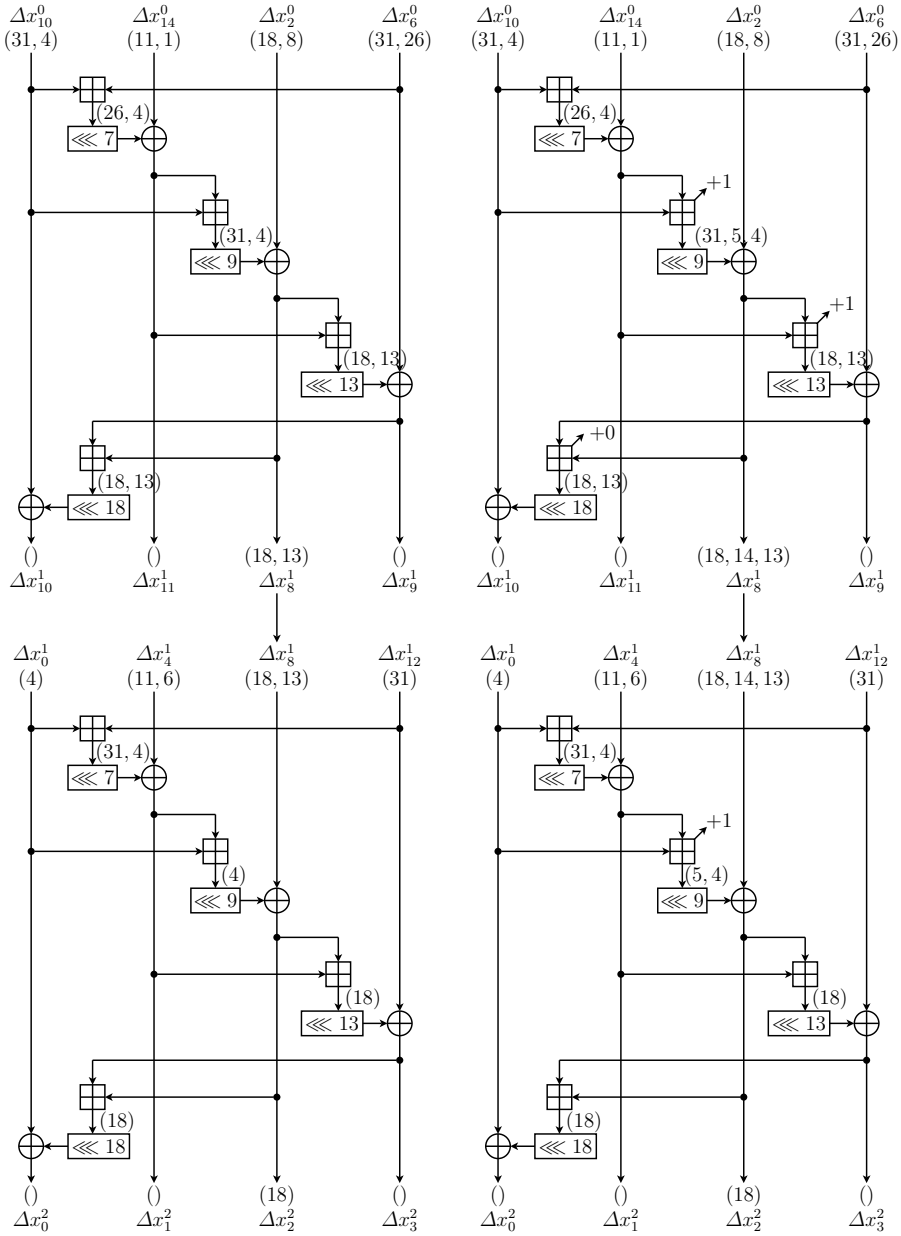

Fig. 3. Differential B (defined in Sect. 5.3) is composed of several characteristics. We show the two with the lowest weight: weight 24 (left), and weight 27 (right). For the characteristic on the right, we use '+1' and '-1' to denote the increase or decrease of weight, compared to the characteristic on the left. In Sect. 5.4, we show that not all operations are independent for the characteristic on the right: this explains why one addition has '+0' instead of '+1'. Note that only part of the characteristics are shown.

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

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

Let us now revisit Differential A. In 2 (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. 3. Here, we see that the characteristic of Fig. 3 (right) has a weight of 27 instead of 28.

The effect is also not limited to cases where a differential consists of several characteristics. 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 characteristic, which are shown in Fig. 4 (left) and Fig. 4 (right) respectively. As shown by Fig. 4 (left), Differential C has a probability of 20 instead of 21 when an analysis of signed differences is taken into account. In Fig. 4 (right), the characteristic with weight 22 turns out to be impossible due to a contradiction.

5.5 Experimental Verification of the Probabilities

In this section, we evaluate the probabilities of three-round Salsa20 characteristic experimentally. We evaluate the probability of Differentials A and B of Sect. 5.3), and Differentials C and D of Sect. 5.4.

We also selected 9 differentials at random for three rounds of Salsa20, one for every weight from 18 up to 26. These characteristics are chosen at random

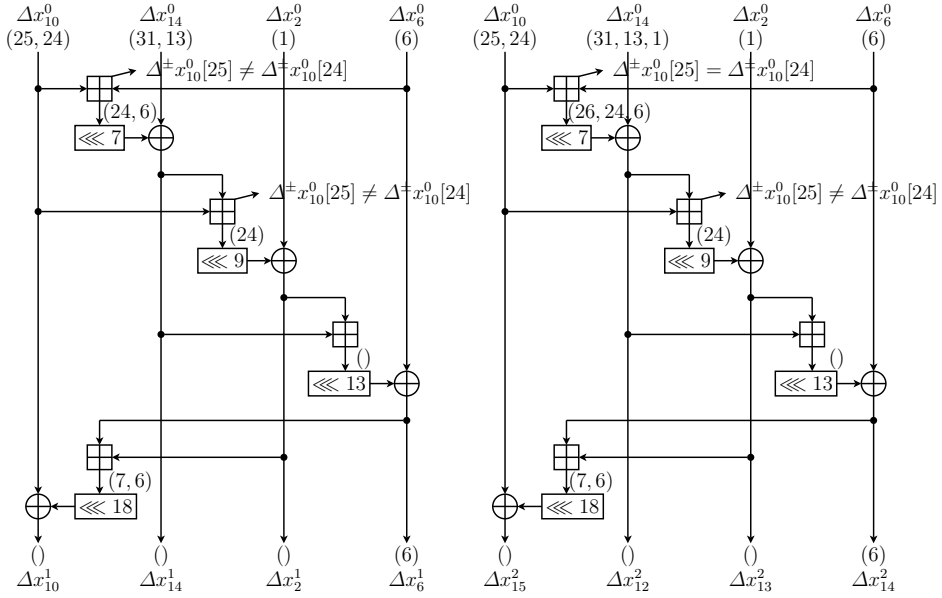

Fig. 4. Differential C (defined in Sect. 5.4) consists of exactly one characteristic (left), with a weight of 20 instead of weight 21 due to the condition $\Delta^{\pm} x_{10}^0[25] \neq \Delta^{\pm} x_{10}^0[24]$. Differential D (see Sect. 5.4) also consists of one characteristic (right). Although a first estimate showed that this characteristic had a weight of 21, we actually find that the characteristic 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 characteristics are shown.

with the following requirements: $\Delta x_0^0[31] = 0$, there is only one characteristic 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 probability one. These differentials can be found in Sect. A.2.

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 3. 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.

5.6 Discussion

Our analysis uses conditions on two bits to verify characteristics. Two-bit conditions have been used before for the manual cryptanalysis of the members of the MD4 family by Wang et al. [38], and subsequently in automated techniques by Mouha et al. [27] and Mendel et al. [26]. These two-bit conditions mostly occur due to the propagation of differences in the Boolean functions.

Table 3. Experimental Probabilities of Differentials for Three Rounds of Salsa20.

Name	Ref.	Theoretical Weight	Sample Size	Expected Value	Experimental Value	Difference	p-value
Differential 18	§ A.2	18	2^{34}	65,536	65,800	+264	0.3033
Differential 19	§ A.2	19	2^{35}	65,536	65,209	-327	0.2022
Differential 20	§ A.2	20	2^{36}	65,536	65,496	-40	0.8774
Differential 21	§ A.2	21	2^{37}	65,536	65,664	+128	0.6185
Differential 22	§ A.2	22	2^{38}	65,536	65,667	+131	0.6102
Differential 23	§ A.2	23	2^{39}	65,536	65,119	-417	0.1037
Differential 24	§ A.2	24	2^{40}	65,536	65,725	+189	0.4615
Differential 25	§ A.2	25	2^{41}	65,536	65,113	-423	0.0989
Differential 26	§ A.2	26	2^{42}	65,536	65,848	+312	0.2237
Differential A	§ 5.3	23.81	2^{40}	74,896	75,227	+331	0.2272
Differential B	§ 5.3	24.42	2^{41}	98,304	98,264	-40	0.8997
Differential C	§ 5.4	20	2^{36}	65,536	65,453	-83	0.7473
Differential D	§ 5.4	infinity	2^{42}	0	0	0	1

For ARX ciphers, Leurent proposed conditions on two and on three adjacent bits of one word in [22]. Although these conditions capture the conditions that encounter in Sect. 5.3 and Sect. 5.4, they do not detect the necessary conditions that appear in Salsa20 characteristics.

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.

In the case of BLAKE and Skein, the output of an addition can be used directly as the input of another addition. As can be seen from Fig. 1, this situation never occurs in Salsa20. For Salsa20, the output of the addition of two variables is, after rotation, XORed with a third variable. This XOR operation has the effect of *whitening* the outputs, i.e. to ensure that for a particular output difference, the values of the output pairs are uniformly distributed. This assumption greatly simplifies our analysis, and seems to seems to be valid since we consider very sparse characteristics. However, this property needs to be investigated further in future work.

Note that this assumption of uniformly distributed outputs is seemingly contradicted by Leurent's ARXtools [22]. This tool finds that for the last characteristic of App. A.1, bits $(x_{13}^2[21], x_{13}^2[20], x_{13}^2[20])$ cannot have values $(1, 0, 0)$ or $(0, 1, 1)$. However, this does not imply that the x_{13}^2 is not uniformly distributed when it is output by the XOR operation. It can be seen that these conditions are the consequence of subsequent operations. By removing all subsequent operations that use x_{13}^2 as inputs and running ARXtools again, we find that $(x_{13}^2[21], x_{13}^2[20], x_{13}^2[20])$ can be assigned to all 2^8 values.

We also note that the contradiction found by Leurent [22] for the near-collision attack on step-reduced Skein-256 by Yu et al. [42] does not occur for our characteristics. The reason is that this inconsistency is caught by the Lipmaa-Moriai formula of Sect. 4.1.

6 Conclusion and Future Work

In this paper, we provide a toolkit to efficiently find optimal characteristics for ARX ciphers, assuming that the probability of a characteristic is equal to the product of the probability of its component. The same technique can also be used to find all characteristics corresponding to a given differential.

The framework that we propose, requires writing equations for every addition, rotation and XOR of the ARX cipher. For the addition operation, we use the formulae proposed by Lipmaa and Moriai to restrict the input and output differences to valid differentials, and to calculate the differential probability. Our framework then uses a SAT solver to find characteristics up to a certain weight.

We perform extensive experiments for three rounds of Salsa20. We find that there are 6,761,988 differentials for weights of up to 26. Out of these, only 2,604 differentials consist of more than one characteristic, and this reduces the weight of the differentials only slightly.

It is commonly assumed for ARX ciphers that the probability of a characteristic can be computed by multiplying the probability of each component. However, we find that this analysis does not hold for certain characteristics. In order to resolve this issue, we introduce conditions between two (not necessarily adjacent) bits of one word, to indicate whether their signed differences are opposite or equal.

We then perform a extensive amount of experiments to verify that the experimental probability of the differentials matches the theoretical probability that we assumed.

The integration of these signed differences into the automatic search for characteristics is an interesting topic for future research.

Acknowledgments. The authors would like to thank (in alphabetical order) Dong-Chan Kim, Ethan Heilman, Deukjo Hong, Atul Luykx, Nikos Mavrogiannopoulos, Florian Mendel, Vincent Rijmen and Kan Yasuda for their useful comments and suggestions.

References

1. Aumasson, J.P., Çagdas Çalik, Meier, W., Özen, O., Phan, R.C.W., Varıcı, K.: Improved Cryptanalysis of Skein. In: Matsui, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 542–559. Springer (2009)
2. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg [29], pp. 470–488

3. Aumasson, J.P., Guo, J., Knellwolf, S., Matusiewicz, K., Meier, W.: Differential and Invertibility Properties of BLAKE. In: Hong, S., Iwata, T. (eds.) FSE. Lecture Notes in Computer Science, vol. 6147, pp. 318–332. Springer (2010)
4. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to the NIST SHA-3 Competition (Round 3) (2010), http://131002.net/blake/blake.pdf
5. Bernstein, D.J.: Salsa20 specification. http://cr.yp.to/snuffle/spec.pdf (April 2005)
6. Bernstein, D.J.: Salsa20/8 and Salsa20/12. http://cr.yp.to/snuffle/812.pdf (February 2006)
7. Bernstein, D.J.: Response to "On the Salsa20 core function". http://cr.yp.to/snuffle/reoncore-20080224.pdf (February 2008)
8. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw and Billet [30], pp. 84–97
9. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
10. Biryukov, A., Nikolic, I., Roy, A.: Boomerang Attacks on BLAKE-32. In: Joux, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 6733, pp. 218–237. Springer (2011)
11. 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 Transactions on Information Theory 44(1), 367–378 (1998)
12. Castro, J.C.H., Estévez-Tapiador, J.M., Quisquater, J.J.: On the Salsa20 Core Function. In: Nyberg [29], pp. 462–469
13. Cramer, R. (ed.): Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3494. Springer (2005)
14. Crowley, P.: Truncated differential cryptanalysis of five rounds of Salsa20. SASC 2006 Workshop: Stream Ciphers Revisted. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/073 (2005), http://www.ecrypt.eu.org/stream
15. Daemen, J., Govaerts, R., Vandewalle, J.: Resynchronization Weaknesses in Synchronous Stream Ciphers. In: EUROCRYPT. pp. 159–167 (1993)
16. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
17. Damm, W., Hermanns, H. (eds.): Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4590. Springer (2007)
18. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 4284, pp. 1–20. Springer (2006)
19. 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
20. Ganesh, V., Dill, D.L.: A Decision Procedure for Bit-Vectors and Arrays. In: Damm and Hermanns [17], pp. 519–531
21. Lai, X., Massey, J.L.: Markov Ciphers and Differentail Cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer (1991)

22. Leurent, G.: Analysis of Differential Attacks in ARX Constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 7658, pp. 226–243. Springer (2012)
23. Leurent, G.: Construction of Differential Characteristics in ARX Designs Application to Skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 8042, pp. 241–258. Springer (2013)
24. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE. Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer (2001)
25. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: EUROCRYPT. pp. 81–91 (1992)
26. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 7073, pp. 288–307. Springer (2011)
27. Mouha, N., De Cannière, C., Indesteege, S., Preneel, B.: Finding Collisions for a 45-Step Simplified HAS-V. In: Youm, H.Y., Yung, M. (eds.) WISA. Lecture Notes in Computer Science, vol. 5932, pp. 206–225. Springer (2009)
28. Nad, T.: The CodingTool Library. In: Workshop on Tools for Cryptanalysis. pp. 129–130 (2010)
29. 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)
30. Robshaw, M.J.B., Billet, O. (eds.): New Stream Cipher Designs - The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986. Springer (2008)
31. Shimizu, A., Miyaguchi, S.: Fast Data Encipherment Algorithm FEAL. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 304, pp. 267–278. Springer (1987)
32. Soos, M.: CryptoMiniSat2. `http://www.msoos.org/cryptominisat2/` (2013)
33. Stevens, M.: On Collisions for MD5. Master's thesis, Eindhoven University of Technology (2007)
34. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4515, pp. 1–22. Springer (2007)
35. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In: Halevi, S. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 5677, pp. 55–69. Springer (2009)
36. Su, B., Wu, W., Wu, S., Dong, L.: Near-Collisions on the Reduced-Round Compression Functions of Skein and BLAKE. In: Heng, S.H., Wright, R.N., Goi, B.M. (eds.) CANS. Lecture Notes in Computer Science, vol. 6467, pp. 124–139. Springer (2010)
37. Wagner, D.: Re-rolled Salsa20 function. `http://groups.google.com/group/sci.crypt/msg/0692e3aaf78687a3` (September 2005)
38. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer [13], pp. 1–18
39. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer (2005)
40. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer [13], pp. 19–35

41. Weinmann, R.P.: AXR - Crypto Made from Modular Additions, XORs and Word Rotations. Dagstuhl Seminar 09031 (January 2009), http://www.dagstuhl.de/Materials/AbstractListing/index.en.phtml?09031
42. Yu, H., Chen, J., Jia, K., Wang, X.: Near-Collision Attack on the Step-Reduced Compression Function of Skein-256. IACR Cryptology ePrint Archive 2011, 148 (2011)
43. Yu, H., Chen, J., Wang, X.: The Boomerang Attacks on the Round-Reduced Skein-512. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 287–303. Springer (2012)

A Additional Supporting Information

A.1 Differential Characteristics for Three Rounds of Salsa20

There are exactly eight three-round differential characteristics for Salsa20 with a weight of less than or equal to 18. Below, we give four of these characteristics. The characteristics show the symmetry of the Salsa20 core function, as explained in [5, 7]. The other four characteristics can be obtained by flipping the MSB of every XOR word difference.

– Characteristic 1:

$$\begin{bmatrix} (31) & (6) & (8,3) & () \\ (6) & (24) & (21) & () \\ (8) & (31,13) & (26) & () \\ () & (1) & (28,1) & () \end{bmatrix} \rightarrow \begin{bmatrix} (31) & () & () & () \\ (6) & () & () & () \\ (8) & () & () & () \\ () & () & () & () \end{bmatrix} \rightarrow \begin{bmatrix} (31) & () & () & () \\ () & () & () & () \\ () & () & () & () \\ () & () & () & () \end{bmatrix} \rightarrow$$
$$\begin{bmatrix} (31,26,14,7,5,1) & (6) & (15,8) & (28,21,19) \\ () & () & () & () \\ () & () & () & () \\ () & () & () & () \end{bmatrix}$$

– Characteristic 2:

$$\begin{bmatrix} () & () & (1) & (28,1) \\ () & (31) & (6) & (8,3) \\ () & (6) & (24) & (21) \\ () & (8) & (31,13) & (26) \end{bmatrix} \rightarrow \begin{bmatrix} () & () & () & () \\ () & (31) & () & () \\ () & (6) & () & () \\ () & (8) & () & () \end{bmatrix} \rightarrow \begin{bmatrix} () & () & () & () \\ () & (31) & () & () \\ () & () & () & () \\ () & () & () & () \end{bmatrix} \rightarrow$$
$$\begin{bmatrix} () & () & () & () \\ (28,21,19) & (31,26,14,7,5,1) & (6) & (15,8) \\ () & () & () & () \\ () & () & () & () \end{bmatrix}$$

– Characteristic 3:

$$\begin{bmatrix} (26) & () & (8) & (31,13) \\ (28,1) & () & () & (1) \\ (8,3) & () & (31) & (6) \\ (21) & () & (6) & (24) \end{bmatrix} \rightarrow \begin{bmatrix} () & () & (8) & () \\ () & () & () & () \\ () & () & (31) & () \\ () & () & (6) & () \end{bmatrix} \rightarrow \begin{bmatrix} () & () & () & () \\ () & () & () & () \\ () & () & (31) & () \\ () & () & () & () \end{bmatrix} \rightarrow$$
$$\begin{bmatrix} () & () & () & () \\ () & () & () & () \\ (15,8) & (26,21,19) & (31,26,14,7,5,1) & (6) \\ () & () & () & () \end{bmatrix}$$

– Characteristic 4:

$$\begin{bmatrix} (24) & (21) & () & (6) \\ (31,13) & (26) & () & (8) \\ (1) & (28,1) & () & () \\ (6) & (8,3) & () & (31) \end{bmatrix} \rightarrow \begin{bmatrix} () & () & () & (6) \\ () & () & () & (8) \\ () & () & () & () \\ () & () & () & (31) \end{bmatrix} \rightarrow \begin{bmatrix} () & () & () & () \\ () & () & () & () \\ () & () & () & () \\ () & () & () & (31) \end{bmatrix} \rightarrow$$
$$\begin{bmatrix} () & () & () & () \\ () & () & () & () \\ () & () & () & () \\ (6) & (15,8) & (28,21,19) & (31,26,14,7,5,1) \end{bmatrix}$$

A.2 Randomly Selected Differentials for Salsa20

In this Section, we give a list of 9 randomly selected differentials for three rounds of Salsa20, one for every weight from 18 up to 26. These differentials were chosen randomly out of the set of all 6,761,988 possible differentials, with the restrictions that $\Delta x_0^0[31] = 0$, that there is only one characteristic per differential, and that no extra conditions appear due to modular differences. For other differentials, we refer to Sect. 5.3 and Sect. 5.4.

We give XOR input and output differences of differentials (in hexadecimal format) according to the following notation: Format:

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

Differential 18, weight 18:

$$\begin{bmatrix} \texttt{01000000} & \texttt{00200000} & \texttt{00000000} & \texttt{00000040} \\ \texttt{80002000} & \texttt{04000000} & \texttt{00000000} & \texttt{00000100} \\ \texttt{00000002} & \texttt{10000002} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00000108} & \texttt{00000000} & \texttt{80000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00008100} & \texttt{10280000} & \texttt{840040a2} \end{bmatrix}$$

Differential 19, weight 19:

$$\begin{bmatrix} \texttt{04000000} & \texttt{00000000} & \texttt{00000100} & \texttt{80006000} \\ \texttt{10000002} & \texttt{00000000} & \texttt{00000000} & \texttt{00000002} \\ \texttt{00000108} & \texttt{00000000} & \texttt{80000000} & \texttt{00000040} \\ \texttt{00200000} & \texttt{00000000} & \texttt{00000040} & \texttt{01000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00008100} & \texttt{10280000} & \texttt{840040a2} & \texttt{00000040} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \end{bmatrix}$$

Differential 20, weight 20:

$$\begin{bmatrix} \texttt{01000000} & \texttt{00200000} & \texttt{00000000} & \texttt{00000040} \\ \texttt{80002001} & \texttt{04000000} & \texttt{00000000} & \texttt{00000100} \\ \texttt{00000002} & \texttt{10000006} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00000108} & \texttt{00000000} & \texttt{80000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00008100} & \texttt{10280000} & \texttt{840040a2} \end{bmatrix}$$

Differential 21, weight 21:

$$\begin{bmatrix} \texttt{01000000} & \texttt{00200000} & \texttt{00000000} & \texttt{00000040} \\ \texttt{80006000} & \texttt{04000000} & \texttt{00000000} & \texttt{00000100} \\ \texttt{00000006} & \texttt{10000002} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00000108} & \texttt{00000000} & \texttt{80000000} \end{bmatrix} \rightarrow \begin{bmatrix} \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000000} & \texttt{00000000} & \texttt{00000000} & \texttt{00000000} \\ \texttt{00000040} & \texttt{00008100} & \texttt{10180000} & \texttt{84004022} \end{bmatrix}$$

Differential 22, weight 22:

$$\begin{bmatrix} 01000000 & 00600000 & 00000000 & 00000040 \\ 80002000 & 04000000 & 00000000 & 00000100 \\ 00000002 & 30000002 & 00000000 & 00000000 \\ 00000040 & 00000118 & 00000000 & 80000000 \end{bmatrix} \rightarrow \begin{bmatrix} 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000040 & 00008100 & 10280000 & 840041a2 \end{bmatrix}$$

Differential 23, weight 23:

$$\begin{bmatrix} 00000000 & 80000040 & 80000118 & 80000000 \\ 80000040 & 81000000 & 80200000 & 80000000 \\ 80000100 & 00002000 & 84000000 & 80000000 \\ 80000000 & 80000002 & 90000002 & 80000000 \end{bmatrix} \rightarrow \begin{bmatrix} 0401c066 & 80000040 & 80008100 & 90280000 \\ 80000000 & 80000000 & 80000000 & 80000000 \\ 80000000 & 80000000 & 80000000 & 80000000 \\ 80000000 & 80000000 & 80000000 & 80000000 \end{bmatrix}$$

Differential 24, weight 24:

$$\begin{bmatrix} 04000000 & 00000000 & 00000100 & 80002001 \\ 30000002 & 00000000 & 00000000 & 00000006 \\ 00000108 & 00000000 & 80000000 & 00000040 \\ 00200000 & 00000000 & 00000040 & 01000000 \end{bmatrix} \rightarrow \begin{bmatrix} 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00008100 & 10580000 & 84004162 & 00000040 \\ 00000000 & 00000000 & 00000000 & 00000000 \end{bmatrix}$$

Differential 25, weight 25:

$$\begin{bmatrix} 00000000 & 00000000 & 00000006 & 10000002 \\ 00000000 & 80000000 & 00000040 & 00000118 \\ 00000000 & 00000040 & 03000000 & 00200000 \\ 00000000 & 00000100 & 80002000 & 04000000 \end{bmatrix} \rightarrow \begin{bmatrix} 00000000 & 00000000 & 00000000 & 00000000 \\ 10280000 & 8403c1a2 & 00000040 & 00008100 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \end{bmatrix}$$

Differential 26, weight 26:

$$\begin{bmatrix} 00000000 & 00000000 & 08000000 & 08400000 \\ 00000000 & 02000000 & 00000001 & 20000004 \\ 00000000 & 00000001 & 00040000 & 00008000 \\ 00000000 & 00000004 & 02000080 & 00100000 \end{bmatrix} \rightarrow \begin{bmatrix} 00000000 & 00000000 & 00000000 & 00000000 \\ 00406000 & 8a100301 & 00000001 & 00000204 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \end{bmatrix}$$

A.3 Source Code to find Characteristics for Three Rounds of Salsa20

```
#!/usr/bin/perl -w

$string = <<'HERE';

% inputs + outputs
x00, x01, x02, x03, x04, x05, x06, x07,
x08, x09, x10, x11, x12, x13, x14, x15: BITVECTOR(32);
y00, y01, y02, y03, y04, y05, y06, y07,
y08, y09, y10, y11, y12, y13, y14, y15: BITVECTOR(32);
z00, z01, z02, z03, z04, z05, z06, z07,
z08, z09, z10, z11, z12, z13, z14, z15: BITVECTOR(32);
u00, u01, u02, u03, u04, u05, u06, u07,
u08, u09, u10, u11, u12, u13, u14, u15: BITVECTOR(32);
```

```
14
15 % output of addition
16 c00, c01, c02, c03, c04, c05, c06, c07,
17 c08, c09, c10, c11, c12, c13, c14, c15 : BITVECTOR(32);
18 c16, c17, c18, c19, c20, c21, c22, c23,
19 c24, c25, c26, c27, c28, c29, c30, c31 : BITVECTOR(32);
20 c32, c33, c34, c35, c36, c37, c38, c39,
21 c40, c41, c42, c43, c44, c45, c46, c47 : BITVECTOR(32);
22
23 % helper variable to calculate weight
24 p00, p01, p02, p03, p04, p05, p06, p07,
25 p08, p09, p10, p11, p12, p13, p14, p15 : BITVECTOR(32);
26 p16, p17, p18, p19, p20, p21, p22, p23,
27 p24, p25, p26, p27, p28, p29, p30, p31 : BITVECTOR(32);
28 p32, p33, p34, p35, p36, p37, p38, p39,
29 p40, p41, p42, p43, p44, p45, p46, p47 : BITVECTOR(32);
30
31 % weight of the characteristic
32 weight : BITVECTOR(11);
33
34 % first round
35 ASSERT( add(x00,x12,c00) );
36 ASSERT( (y04 = BVXOR(x04, rotl(c00,7))) );
37 ASSERT( pen(x00,x12,c00,p00) );
38
39 ASSERT( add(y04,x00,c01) );
40 ASSERT( (y08 = BVXOR(x08, rotl(c01,9))) );
41 ASSERT( pen(y04,x00,c01,p01) );
42
43 ASSERT( add(y08,y04,c02) );
44 ASSERT( (y12 = BVXOR(x12, rotl(c02,13))) );
45 ASSERT( pen(y08,y04,c02,p02) );
46
47 ASSERT( add(y12,y08,c03) );
48 ASSERT( (y00 = BVXOR(x00, rotl(c03,18))) );
49 ASSERT( pen(y12,y08,c03,p03) );
50
51 ASSERT( add(x05,x01,c04) );
52 ASSERT( (y09 = BVXOR(x09, rotl(c04,7))) );
53 ASSERT( pen(x05,x01,c04,p04) );
54
55 ASSERT( add(y09,x05,c05) );
56 ASSERT( (y13 = BVXOR(x13, rotl(c05,9))) );
57 ASSERT( pen(y09,x05,c05,p05) );
58
59 ASSERT( add(y13,y09,c06) );
60 ASSERT( (y01 = BVXOR(x01, rotl(c06,13))) );
61 ASSERT( pen(y13,y09,c06,p06) );
62
63 ASSERT( add(y01,y13,c07) );
```

```
64 ASSERT( (y05 = BVXOR(x05, rotl(c07,18))) );
65 ASSERT( pen(y01,y13,c07,p07) );
66
67 ASSERT( add(x10,x06,c08) );
68 ASSERT( (y14 = BVXOR(x14, rotl(c08,7))) );
69 ASSERT( pen(x10,x06,c08,p08) );
70
71 ASSERT( add(y14,x10,c09) );
72 ASSERT( (y02 = BVXOR(x02, rotl(c09,9))) );
73 ASSERT( pen(y14,x10,c09,p09) );
74
75 ASSERT( add(y02,y14,c10) );
76 ASSERT( (y06 = BVXOR(x06, rotl(c10,13))) );
77 ASSERT( pen(y02,y14,c10,p10) );
78
79 ASSERT( add(y06,y02,c11) );
80 ASSERT( (y10 = BVXOR(x10, rotl(c11,18))) );
81 ASSERT( pen(y06,y02,c11,p11) );
82
83 ASSERT( add(x15,x11,c12) );
84 ASSERT( (y03 = BVXOR(x03, rotl(c12,7))) );
85 ASSERT( pen(x15,x11,c12,p12) );
86
87 ASSERT( add(y03,x15,c13) );
88 ASSERT( (y07 = BVXOR(x07, rotl(c13,9))) );
89 ASSERT( pen(y03,x15,c13,p13) );
90
91 ASSERT( add(y07,y03,c14) );
92 ASSERT( (y11 = BVXOR(x11, rotl(c14,13))) );
93 ASSERT( pen(y07,y03,c14,p14) );
94
95 ASSERT( add(y11,y07,c15) );
96 ASSERT( (y15 = BVXOR(x15, rotl(c15,18))) );
97 ASSERT( pen(y11,y07,c15,p15) );
98
99 % second round
100 ASSERT( add(y00,y03,c16) );
101 ASSERT( (z01 = BVXOR(y01, rotl(c16,7))) );
102 ASSERT( pen(y00,y03,c16,p16) );
103
104 ASSERT( add(z01,y00,c17) );
105 ASSERT( (z02 = BVXOR(y02, rotl(c17,9))) );
106 ASSERT( pen(z01,y00,c17,p17) );
107
108 ASSERT( add(z02,z01,c18) );
109 ASSERT( (z03 = BVXOR(y03, rotl(c18,13))) );
110 ASSERT( pen(z02,z01,c18,p18) );
111
112 ASSERT( add(z03,z02,c19) );
113 ASSERT( (z00 = BVXOR(y00, rotl(c19,18))) );
```

```
114 ASSERT( pen(z03,z02,c19,p19) );
115
116 ASSERT( add(y05,y04,c20) );
117 ASSERT( (z06 = BVXOR(y06, rotl(c20,7))) );
118 ASSERT( pen(y05,y04,c20,p20) );
119
120 ASSERT( add(z06,y05,c21) );
121 ASSERT( (z07 = BVXOR(y07, rotl(c21,9))) );
122 ASSERT( pen(z06,y05,c21,p21) );
123
124 ASSERT( add(z07,z06,c22) );
125 ASSERT( (z04 = BVXOR(y04, rotl(c22,13))) );
126 ASSERT( pen(z07,z06,c22,p22) );
127
128 ASSERT( add(z04,z07,c23) );
129 ASSERT( (z05 = BVXOR(y05, rotl(c23,18))) );
130 ASSERT( pen(z04,z07,c23,p23) );
131
132 ASSERT( add(y10,y09,c24) );
133 ASSERT( (z11 = BVXOR(y11, rotl(c24,7))) );
134 ASSERT( pen(y10,y09,c24,p24) );
135
136 ASSERT( add(z11,y10,c25) );
137 ASSERT( (z08 = BVXOR(y08, rotl(c25,9))) );
138 ASSERT( pen(z11,y10,c25,p25) );
139
140 ASSERT( add(z08,z11,c26) );
141 ASSERT( (z09 = BVXOR(y09, rotl(c26,13))) );
142 ASSERT( pen(z08,z11,c26,p26) );
143
144 ASSERT( add(z09,z08,c27) );
145 ASSERT( (z10 = BVXOR(y10, rotl(c27,18))) );
146 ASSERT( pen(z09,z08,c27,p27) );
147
148 ASSERT( add(y15,y14,c28) );
149 ASSERT( (z12 = BVXOR(y12, rotl(c28,7))) );
150 ASSERT( pen(y15,y14,c28,p28) );
151
152 ASSERT( add(z12,y15,c29) );
153 ASSERT( (z13 = BVXOR(y13, rotl(c29,9))) );
154 ASSERT( pen(z12,y15,c29,p29) );
155
156 ASSERT( add(z13,z12,c30) );
157 ASSERT( (z14 = BVXOR(y14, rotl(c30,13))) );
158 ASSERT( pen(z13,z12,c30,p30) );
159
160 ASSERT( add(z14,z13,c31) );
161 ASSERT( (z15 = BVXOR(y15, rotl(c31,18))) );
162 ASSERT( pen(z14,z13,c31,p31) );
163
```

```
% third round
ASSERT( add(z00,z12,c32) );
ASSERT( (u04 = BVXOR(z04, rotl(c32,7))) );
ASSERT( pen(z00,z12,c32,p32) );

ASSERT( add(u04,z00,c33) );
ASSERT( (u08 = BVXOR(z08, rotl(c33,9))) );
ASSERT( pen(u04,z00,c33,p33) );

ASSERT( add(u08,u04,c34) );
ASSERT( (u12 = BVXOR(z12, rotl(c34,13))) );
ASSERT( pen(u08,u04,c34,p34) );

ASSERT( add(u12,u08,c35) );
ASSERT( (u00 = BVXOR(z00, rotl(c35,18))) );
ASSERT( pen(u12,u08,c35,p35) );

ASSERT( add(z05,z01,c36) );
ASSERT( (u09 = BVXOR(z09, rotl(c36,7))) );
ASSERT( pen(z05,z01,c36,p36) );

ASSERT( add(u09,z05,c37) );
ASSERT( (u13 = BVXOR(z13, rotl(c37,9))) );
ASSERT( pen(u09,z05,c37,p37) );

ASSERT( add(u13,u09,c38) );
ASSERT( (u01 = BVXOR(z01, rotl(c38,13))) );
ASSERT( pen(u13,u09,c38,p38) );

ASSERT( add(u01,u13,c39) );
ASSERT( (u05 = BVXOR(z05, rotl(c39,18))) );
ASSERT( pen(u01,u13,c39,p39) );

ASSERT( add(z10,z06,c40) );
ASSERT( (u14 = BVXOR(z14, rotl(c40,7))) );
ASSERT( pen(z10,z06,c40,p40) );

ASSERT( add(u14,z10,c41) );
ASSERT( (u02 = BVXOR(z02, rotl(c41,9))) );
ASSERT( pen(u14,z10,c41,p41) );

ASSERT( add(u02,u14,c42) );
ASSERT( (u06 = BVXOR(z06, rotl(c42,13))) );
ASSERT( pen(u02,u14,c42,p42) );

ASSERT( add(u06,u02,c43) );
ASSERT( (u10 = BVXOR(z10, rotl(c43,18))) );
ASSERT( pen(u06,u02,c43,p43) );

ASSERT( add(z15,z11,c44) );
```

```
214 ASSERT( (u03 = BVXOR(z03, rotl(c44,7))) );
215 ASSERT( pen(z15,z11,c44,p44) );
216
217 ASSERT( add(u03,z15,c45) );
218 ASSERT( (u07 = BVXOR(z07, rotl(c45,9))) );
219 ASSERT( pen(u03,z15,c45,p45) );
220
221 ASSERT( add(u07,u03,c46) );
222 ASSERT( (u11 = BVXOR(z11, rotl(c46,13))) );
223 ASSERT( pen(u07,u03,c46,p46) );
224
225 ASSERT( add(u11,u07,c47) );
226 ASSERT( (u15 = BVXOR(z15, rotl(c47,18))) );
227 ASSERT( pen(u11,u07,c47,p47) );
228
229 % Flipping MSBs: also a valid characteristic:
230 ASSERT( x00[31:31] = 0b0 );
231
232 % No all-zero characteristic:
233 ASSERT( NOT((x00|x01|x02|x03|x04|x05|x06|x07|x08|x09|x10|x11|
      x12|x13|x14|x15) = 0hex00000000) );
234
235 ASSERT( (weight = (BVPLUS(11,0b000000@wtx(p00),
236   0b000000@wtx(p01), 0b000000@wtx(p02), 0b000000@wtx(p03),
237   0b000000@wtx(p04), 0b000000@wtx(p05), 0b000000@wtx(p06),
238   0b000000@wtx(p07), 0b000000@wtx(p08), 0b000000@wtx(p09),
239   0b000000@wtx(p10), 0b000000@wtx(p11), 0b000000@wtx(p12),
240   0b000000@wtx(p13), 0b000000@wtx(p14), 0b000000@wtx(p15),
241   0b000000@wtx(p16), 0b000000@wtx(p17), 0b000000@wtx(p18),
242   0b000000@wtx(p19), 0b000000@wtx(p20), 0b000000@wtx(p21),
243   0b000000@wtx(p22), 0b000000@wtx(p23), 0b000000@wtx(p24),
244   0b000000@wtx(p25), 0b000000@wtx(p26), 0b000000@wtx(p27),
245   0b000000@wtx(p28), 0b000000@wtx(p29), 0b000000@wtx(p30),
246   0b000000@wtx(p31), 0b000000@wtx(p32), 0b000000@wtx(p33),
247   0b000000@wtx(p34), 0b000000@wtx(p35), 0b000000@wtx(p36),
248   0b000000@wtx(p37), 0b000000@wtx(p38), 0b000000@wtx(p39),
249   0b000000@wtx(p40), 0b000000@wtx(p41), 0b000000@wtx(p42),
250   0b000000@wtx(p43), 0b000000@wtx(p44), 0b000000@wtx(p45),
251   0b000000@wtx(p46), 0b000000@wtx(p47))))
252 );
253
254 ASSERT( BVLE(weight, 0b00000010010) );
255
256 QUERY(FALSE);
257
258 COUNTEREXAMPLE;
259
260 HERE
261
```

```
262 sub eval_rot { return "((($1) << $2)[31:0] | (($1) >> " . (32-
    $2) . "))"; }
263
264 $string =~ s/add\((.*?),(.*?),(.*?)\)/(eq(shift1($1),shift1($2
    ),shift1($3)) & (BVXOR(BVXOR(BVXOR(($1),($2)),($3)),shift1
    ($2))) = 0hex00000000)/g; # add(a,b,c) means a + b = c
265 $string =~ s/pen\((.*?),(.*?),(.*?),(.*?)\)/(($4) = (~eq($1,$2
    ,$3)))/g; # pen(a,b,c,p) means p is penalty for a + b = c
266 $string =~ s/rotl\((.*?),(.*?)\)/&eval_rot($1,$2)/eg; # rotate
     left
267 $string =~ s/eq\((.*?),(.*?),(.*?)\)/(BVXOR(~($1),($2)) &
    BVXOR(~($1),($3)))/g; # equals function
268 $string =~ s/shift1\((.*?)\)/((($1) << 1)[31:0])/g; # shift
    the the left by one position
269 $string =~ s/wtx\((.*?)\)/(BVPLUS(5,0bin0000@($1\[0:0]),0
    bin0000@($1\[1:1]),0bin0000@($1\[2:2]),0bin0000@($1\[3:3])
    ,0bin0000@($1\[4:4]),0bin0000@($1\[5:5]),0bin0000@($1
    \[6:6]),0bin0000@($1\[7:7]),0bin0000@($1\[8:8]),0bin0000@(
    $1\[9:9]),0bin0000@($1\[10:10]),0bin0000@($1\[11:11]),0
    bin0000@($1\[12:12]),0bin0000@($1\[13:13]),0bin0000@($1
    \[14:14]),0bin0000@($1\[15:15]),0bin0000@($1\[16:16]),0
    bin0000@($1\[17:17]),0bin0000@($1\[18:18]),0bin0000@($1
    \[19:19]),0bin0000@($1\[20:20]),0bin0000@($1\[21:21]),0
    bin0000@($1\[22:22]),0bin0000@($1\[23:23]),0bin0000@($1
    \[24:24]),0bin0000@($1\[25:25]),0bin0000@($1\[26:26]),0
    bin0000@($1\[27:27]),0bin0000@($1\[28:28]),0bin0000@($1
    \[29:29]),0bin0000@($1\[30:30])))/g; # hamming weight,
    minus MSB
270
271 print $string;
```

Listing 1.1. Tool to Find Characteristics for Three Rounds of Salsa20 with Weight up to 18, to execute: `perl lemma1.pl | stp`