A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis*,**

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, not a single ARX cipher has yet been proven to be secure against one of the most common attacks in symmetric-key cryptography: differential cryptanalysis. In this paper, we prove that no differential characteristic exists for 15 rounds of Salsa20 with a higher probability than 2^{-130}. Thereby, we show that the full 20-round Salsa20 with a 128-bit key is secure against differential cryptanalysis, with a security margin of 5 rounds. Our proof holds both in single-key and related-key settings. Furthermore, our proof technique only involves writing out simple equations for every addition, rotation and XOR operation in the cipher, and applying an off-the-shelf SAT solver. To prove that Salsa20 is secure against differential cryptanalysis requires only about 20 hours of computation on a single CPU core.

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

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 [2] and Skein [26], the tweakable block cipher Threefish [26] and the stream cipher Salsa20 [6].

* The framework proposed in this paper is accompanied by a software toolkit, available at: http://www.ecrypt.eu.org/tools/salsa20proof

** This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and by the iMinds-MiX-MediaID project. In addition, this work was supported by the Flemish Government, IWT SBO SPION and IWT SBO MobCom, FWO G.0360.11N Location Privacy and FWO G.068611N Data mining, by the European Commission through the CIP thematic network with contract number 270901 SSEDIC and through the SECURITY program under FP7-SEC-2011-284862 Fidelity.

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 cipher FEAL was published [46].

ARX ciphers are very fast in software. On an Intel Core i7 processor, the ARX stream cipher Salsa20 reaches a speed of 3.23 cycles/byte [7], compared to 6.92 cycles/byte AES-128 [32]. This last figure corresponds to the fastest counter-mode AES-128 implementation, that does not make use of the Intel AES New Instructions (AES-NI) [28]. With AES-NI, AES-128 in counter mode requires only 1.26 cycles/byte on the same processor [7]. However, software developers cannot always assume that AES-NI are present, because these instructions were only introduced very recently. Moreover, Salsa20 also has an advantage over AES on smartphones and tablets: Salsa20 runs at only 5.14 cycles/byte on a Qualcomm Snapdragon S4 processor, compared to 18.62 cycles/byte for AES-128 in counter mode [7].

In certain applications where encryption speed is the bottleneck, ARX ciphers offer a critical speed advantage. An example is VMWare View, which uses PCoIP (PC-over-IP) to transmit computer displays over a network. In VMWare View 4.5 and later, 12-round Salsa20 is used by default because it supports speeds up to about 20 Mbit/s, whereas AES only supports speeds up to about 7 Mbit/s [48]. There is currently also an IETF proposal for the Transport Layer Security (TLS) protocol to support both 12-round and 20-round Salsa20, because of performance issues with currently included ciphers [29].

A serious problem with ARX ciphers, however, is that their security is not well understood. To analyze the security of symmetric-key cryptographic primitives, two of the most powerful techniques are differential cryptanalysis [10] and linear cryptanalysis [36]. 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 [21].

For ARX ciphers, there are currently no proven security bounds against differential cryptanalysis in existing literature. Because ARX ciphers are so fast in software compared to compared to ciphers based on substitution-permutation networks (SPNs), ARX ciphers are typically overdesigned. A very large number of rounds is used, because the optimal trade-off between security and efficiency is not 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

[3] Although Weinmann initially introduced the term AXR [52], he changed this into ARX soon afterwards. We use ARX in this paper, because this term has come into wide acceptance in cryptographic literature.

characteristics, and explaining that high-probability differential characteristics could not be found for a sufficient number of rounds [2, 6, 26].

This paper is the first to prove that an ARX cipher is secure against differential cryptanalysis, by proving an upper bound for the probability for the best differential characteristic. Our proof relies on a computer program that finds the best differential characteristics. This program is very easy to generate, as it only involves writing out simple equations for every addition, rotation and XOR of the ARX cipher. These equations are then input into STP [27], 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. Using this framework, we prove after about 20 hours computation that the stream cipher Salsa20 [6] with a 128-bit key is secure against differential cryptanalysis, both in the single-key and in the related-key model. In particular, we prove that no differential characteristic exists for 15 rounds of Salsa20 with a probability of more than 2^{-130}. Because our techniques are not specific to Salsa20, they can also be applied to other ARX ciphers. We have released our results as a publicly available toolkit: `http://www.ecrypt.eu.org/tools/salsa20proof`

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 construct bounds for differential characteristics of ARX ciphers. This framework is applied to Salsa20 in Sect. 5, where we provide a proof that Salsa20 with 128-bit keys is secure against differential cryptanalysis. Section 6 provides an in-depth discussion of our framework, and explains several considerations related to our Salsa20 proof. We conclude in Sect. 7. For three rounds of Salsa20, there are exactly eight differential characteristics with a probability of up to 2^{-18}. The construction of these characteristics is given in App. A.

2 Related Work

Biham and Shamir introduced differential cryptanalysis in [10]. 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 [51] and SHA-1 [50]. For stream ciphers, differential cryptanalysis can be used in the context of a resynchronization attack [20].

To prove the security of ciphers against differential cryptanalysis, Lai et al. introduced the theory of Markov ciphers [33]. Their paper was the

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': $\Delta x = x \oplus x'$
$x[i]$	bit selection: bit at position i of word x, where $i = 0$ is the least significant bit

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 [21]. 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.

Calculating security bounds for AES against related-key attacks is more difficult, but is possible with a search program. This was done by Biryukov and Nikolić [12] by constructing a dedicated search program for byte-oriented ciphers, and by Mouha et al. using Mixed-Integer Linear Programming (MILP) [37,38]. For other ciphers with S-boxes, security bounds against differential cryptanalysis can also be proven by use of search programs. See for example the security proofs for Generalized Feistel Structures (GFSs) [13, 14, 31, 45].

For ciphers without S-boxes, not many ciphers have a proof of security against differential cryptanalysis. We mention the following notable results:

– To prove the security of differential attacks based on local collisions, Jutla and Patthak introduced SHA1-IME [30], a variant of SHA-1 with an improved different message expansion. Bouillaguet et al. constructed a similar proof for SIMD [15]. However, they give more power to the adversary than in the proof for SHA1-IME.

– Rivest et al. submitted the hash function MD6 [42] to the NIST SHA-3 competition [40]. The only data operations used in MD6 are AND, XOR, left and right shifts. MD6 came was submitted with a proof of resistance against differential cryptanalysis. However, NIST stated that MD6 was not a competitive algorithm in the SHA-3 competition because it is much slower than the SHA-2 family of hash functions. In response, Rivest et al. tried to speed up MD6 while still retaining a proof of security, but their attempts were unsuccessful [43].
– The hash function Keccak [8] was selected as the winner of the SHA-3 competition. All its data operations are linear in GF(2), except for the AND function in the χ mapping. Keccak comes with a proof of security against differential cryptanalysis.
– The stream cipher Trivium [23, 25] is a finalist of the eSTREAM competition, and uses only AND and XOR functions. Recently, Trivium was standardized as ISO/IEC 29192-3. Trivium comes with a proof of security against differential cryptanalysis. The block ciphers KATAN and KTANTAN [24] are based on Trivium, and come with a similar proof against differential cryptanalysis.

In existing literature, not a single ARX cipher has yet been proven secure against differential cryptanalysis. This paper is the first to provide such a proof, more specifically for the stream cipher Salsa20.

3 Description of Salsa20

Salsa20 is a stream cipher designed by Bernstein [6]. 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 [4]. 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, where the cipher has been very successful. 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 either Salsa20/12 or Salsa20/20.

Although eSTREAM has standardized Salsa20 with 12 rounds, Bernstein is more conservative and recommends choosing $R = 20$ for typical cryptographic applications [6].

Salsa20 supports both 128-bit and 256-bit keys. In this paper, we will only consider 128-bit keys. Salsa20 operates on 32-bit words. The Salsa20 core function converts a 128-bit key (k_0, k_1, k_2, k_3), 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 256-bit output. The inputs are mapped to a two-dimensional square matrix as follows:

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

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 [6] 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 Construct Bounds for Differential Characteristics of ARX Ciphers

4.1 Calculating Differential Probabilities

Given a differential characteristic, we want to calculate the probability with which it holds. Throughout this paper, 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 [35], 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 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

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

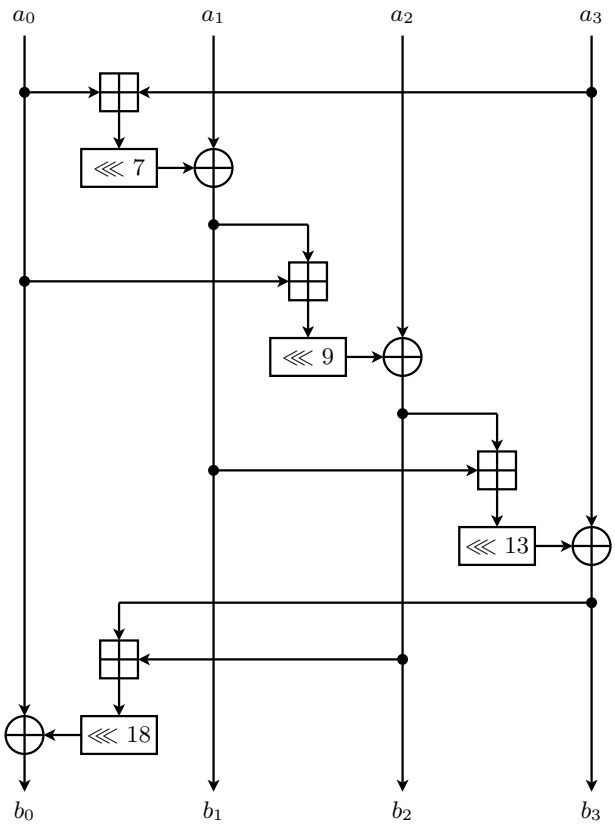

Fig. 1. The Salsa20 `quarterround` function is defined as: $(b_0, b_1, b_2, b_3) \leftarrow \texttt{quarterround}(a_0, a_1, a_2, a_3)$

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:

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

where $h^*(x)$ denotes the number of non-zero bits in x, not counting $x[31]$.

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. 6, we will revisit this assumption and explain why it is valid for our Salsa20 proof.

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 characteristics exists. Our goal will be to make W sufficiently small, so that the search space is limited and the SAT solver terminates within a reasonable time. However, W should also be large enough to derive a useful security bound.

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-sums form of 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, will make use of STP [27]. 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, 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 differences 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 equa-

tions. 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 human errors are made.

5 A Proof that Salsa20 is Secure Against Differential Cryptanalysis

5.1 Applying the Framework to 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 STP. We can then straightforwardly apply the framework of Sect. 4 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, differential characteristics exist with 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 [5], Wagner [49] and Hernandez-Castro et al. [18].

As pointed out by Bernstein [5], 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 $\Delta x_0^0[31] = 0$.

5.2 Results for Salsa20

In order to prove the security of Salsa20 against differential cryptanalysis, we first prove the following lemmata.

Lemma 1. *For three rounds of Salsa20, no differential characteristic exists with a weight of less than 18.*

Proof. 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 [47], 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*, thereby proving that no differential characteristics with a weight of less than 18. □

Lemma 2. *For three rounds of Salsa20, there are exactly eight differential characteristics with a weight of 18.*

Proof. We proceed in a similar way as in Lemma 1, but now use $W = 18$ as a bound. We are interested in finding not just one, but all differential characteristics. Because this 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 not does 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. 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 [3, 5]. □

Lemma 3. *For three rounds of Salsa20, many differential characteristics exist with a weight up to 25. For differentials with $\Delta x_0^0[31] = 0$, there are exactly 286 possible input differences for the third round (Δx_0^2, Δx_1^2,..., Δx_{15}^2).*

Proof. For three rounds of Salsa20, an incomplete search revealed that at least hundreds of thousands of characteristics exist with a weight of up to 25. Adding blocking clauses for each of these solutions greatly increases the number of clauses that the SAT solver has to deal with. This causes the SAT solver to run out of memory.

However, we have noticed that although a very large number of characteristics exist with a weight up to 25, many of these characteristics share the same internal differences. We have constructed an STP program to find characteristics up to $W = 25$, 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 19 hours, 23 minutes and 8 seconds, the SAT solver has found all 286 possible values for the input difference to the third round. □

Lemma 4. *We consider six rounds of Salsa20. The weight of the first three rounds and the last three rounds are respectively denoted by w_1 and w_2. For*

valid characteristics, if $w_1 \leq 25$*, then* $2w_1 + w_2 \geq 78$*. Similarly, if* $w_2 \leq 25$*, then* $w_1 + 2w_2 \geq 78$*.*

Proof. We now construct an STP program to find Salsa20 characteristics for six rounds. If $w_1 \leq 25$ and $\Delta x_0^0[31] = 0$, then we proved in Lemma 3 that there are 286 possible values for Δx_1^2,..., Δx_{15}^2. With these values, we use STP to search for characteristics where $2w_1 + w_2 < 78$. After 10 minutes and 55 seconds, the SAT solver proves that no such characteristics exist. The requirement $\Delta x_0^0[31] = 0$ is again arbitrary: the weight of the characteristics does not change if we flip the MSB of all XOR word differences. Therefore, there are also no characteristics with $\Delta x_0^0[31] = 1$ that satisfy the given criteria.

We similarly prove the case where $w_2 \leq 25$ and $\Delta x_0^3[31] = 0$. After 9 minutes and 43 seconds, the SAT solver proves that if $w_2 \leq 25$, then $w_1 + 2w_2 \geq 78$. □

Theorem 1. *For 15 rounds of Salsa20, no differential characteristic exists with a weight of less than 130.*

Proof. A 15-round Salsa20 characteristic can be split into five parts of three rounds. Let us denote their respective weights by w_1, w_2, ..., w_5.

For every pair of weights (w_i, w_{i+1}) with $1 \leq i < 5$, we can apply Lemma 4 to obtain lower bounds on the weights of differential characteristics.

We input each of these bounds into STP. If we set $w_1 + w_2 + w_3 + w_4 + w_5 < 130$, the program finds after less than a second of computation that no solution exists. This proves that there is no 15-round Salsa20 characteristic with a weight of less than 130. □

Salsa20 consists of 20 rounds, and Theorem 1 proves that the probability of any 15-round differential characteristic is at most 2^{-130}. Therefore, Salsa20 with 128-bit keys is secure against differential cryptanalysis with a security margin of 5 rounds. Our security statement is similar to that of AES, where it was proven that the probability of any four-round differential characteristic is at most 2^{-150} [21].

Because we take into account that differences may be introduced in the key values, our security proof holds both in the single-key and related-key settings.

According to Lemma 3, Lemma 4 and Theorem 1, a 15-round Salsa20 characteristic with weight 130 ($w_1 = w_2 = \ldots = w_5 = 26$) could exist. However, we have not found such a characteristic. Our bound is likely not tight, which means that the security margin of Salsa20 may be even more than five rounds.

6 Considerations

In this section, we explain several considerations that we have taken into account in our research.

Verification of the Search Program. Using our framework to search for differential characteristics, very little programming is required. This significantly reduces the possibility that a human error has been made. Furthermore, we have taken additional several steps to verify our results.

Firstly, we have used two different tools to convert our search programs to CNF format: STP [27] and C32SAT [16]. Both the input languages to these tools, as well as the engines that perform the CNF conversion, differ significantly. Secondly, we have used two different SAT solvers to solve the resulting CNF programs: CryptoMiniSat2 [47] and PicoSAT [9]. Thirdly, the characteristics that were output during the search, were verified by hand calculation.

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 STP less than a second to find all characteristics. For the 3-round characteristics in App. A, we found that there are no other characteristics corresponding to the same differentials.

Characteristics and Probabilities. Recently, Leurent pointed out that several published attacks on ARX ciphers are invalid, because the differential characteristics cannot be satisfied [34]. Leurent's characteristics make use of the *signed differences* that were introduced by Wang et al., which split up the XOR difference $\Delta x[i]$ into three possible cases:

- $x[i] = x'[i]$, which is denoted as 0,
- $x[i] = 0, x'[i] = 1$, which is denoted as +1,
- $x[i] = 1, x'[i] = 0$, which is denoted as -1.

Furthermore, the *multi-bit constraints* in Leurent's characteristics also describe the relations between two adjacent bits: $(x[i], x'[i])$ and $(x[i+1], x'[i+1])$. Using these constraints, Leurent found the characteristics of several published attacks on the SHA-3 finalist hash functions BLAKE [2] and Skein [26] are invalid.

Our analysis of Salsa20 assumes that every output pair of the ARX operations occur with equal probability, as long as the XOR difference is

satisfied. This is clearly not the case when signed differences or multi-bit constraints appear as a result of the addition operation.

In the case of BLAKE and Skein, the output of an addition is 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. The differential characteristics that we consider are very sparse: of the 512-bit input to every round of Salsa20, only very few bits contain a difference. Therefore, the XOR operation has the effect of *whitening* the outputs, which validates our assumption of uniform outputs and allows us to accurately calculate the probability of a differential characteristic by multiplying the probabilities of every ARX operation.

We have performed several experiments to verify that our probability calculations indeed accurate. For example, using the first characteristic of App. A, for 2^{34} pairs of inputs chosen uniformly at random, we found 65620 valid output differences. As the theoretical probability of the characteristic is 2^{-18}, the expected value is $2^{34} \cdot 2^{-18} = 2^{16} = 65536$ valid output differences. The null hypothesis is that the theoretical probability of the characteristic is 2^{-18}. A two-tailed hypothesis test gives a p-value of 0.7443, therefore we accept the null hypothesis. This provides evidence that the experimental probability corresponds to the theoretical probability.

Linearization. Linearization is a common technique to find low-weight characteristics for ARX ciphers. Using this technique, every addition is replaced by XOR, which results in a linear code in GF(2). Standard techniques from coding theory can then be used to find low-weight codewords [17, 39]. Linearization is a very powerful technique, and can find three-round Salsa20 characteristics of weight 18 in only a few seconds.

All three-round characteristics of weight 18 (given in App. A) can be found by linearization. However, to find all linearized characteristics of weight 18 requires more than 19 days of computation on a 2.93GHz Intel Xeon X7350 processor using MAGMA's `MinimumWords` function. Our framework finds the same characteristics in only half a hour. Furthermore, linearization cannot find the non-linear three-round characteristics that exist for weight 19 and higher.

Security Margin. The current best differential attack on Salsa20 is on eight rounds [1]. This attack consists of a four-round differential, and goes back four rounds with partial key guesses. The authors explain that they have done extensive efforts to go back five rounds with partial key guesses, but without success. We note that partial key guesses are much more diffi-

cult in our case, because the result in this paper is on Salsa20 with 128-bit instead of 256-bit keys.

Our proof for Salsa20 has a security margin of five rounds, which should therefore be more than sufficient. Furthermore, this paper does not obtain a tight bound: the best differential characteristic for 15-rounds may have a lower probability than 2^{-130}. The security margin of Salsa20 against differential cryptanalysis may thus be even more than five rounds.

7 Conclusion and Future Work

In this paper, we provided the first proof that an ARX cipher is secure against differential cryptanalysis. In particular, we proved that Salsa20 with the default 20 rounds and with a 128-bit key size is secure against differential cryptanalysis. Our proof is the result of applying a general framework for ARX ciphers, and is not specific to Salsa20.

The framework that we proposed, required writing equations for every addition, rotation and XOR of the ARX cipher. For the addition operation, we used the formulae proposed by Lipmaa and Moriai to restrict the input and output differences to valid differentials, and to calculate the differential probability.

We noted that in Salsa20, the output of the addition of two input variables is always (after rotation) XORed with a third input value. This XOR has the effect of *whitening* the output pairs, therefore for sparse characteristics, it is correct to assume that for a given output difference, the output pairs of an ARX operation are uniformly distributed. We have performed experiments to verify that the experimental and the theoretical probability of Salsa20 characteristics correspond to each other. Note that unlike Salsa20, in several other ARX ciphers such as BLAKE, ChaCha and Skein, the output of an addition is directly input into another addition.

Our proof required a search for the best characteristics, which we performed using a SAT solver. After less than 20 hours of computation on a single CPU core, we proved that no characteristic exists for 15 rounds of Salsa20, with a probability of less than 2^{-130}. As Salsa20 has 20 rounds in total, there is a security margin of five rounds against differential cryptanalysis attacks. Our result holds for 128-bit keys, both in the single-key and in the related-key model.

The framework of this paper allows the designers of ARX ciphers to prove the security of their ciphers against differential cryptanalysis, and therefore make a trade-off between security and efficiency. The security bound provided in this paper is not tight, it is therefore possible to obtain better bounds. The application and improvement of our framework to other ARX ciphers, as well as the extension to linear cryptanalysis, are other interesting topics for future research.

Acknowledgments. The authors would like to thank (in alphabetical order) Atul Luykx, Nikos Mavrogiannopoulos, Florian Mendel, Vincent Rijmen and Kan Yasuda for their useful comments and suggestions.

References

1. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg [41], pp. 470–488
2. 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`
3. Bernstein, D.J.: Salsa20 specification. `http://cr.yp.to/snuffle/spec.pdf` (April 2005)
4. Bernstein, D.J.: Salsa20/8 and Salsa20/12. `http://cr.yp.to/snuffle/812.pdf` (February 2006)
5. Bernstein, D.J.: Response to "On the Salsa20 core function". `http://cr.yp.to/snuffle/reoncore-20080224.pdf` (February 2008)
6. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw and Billet [44], pp. 84–97
7. Bernstein, D.J.: eBACS: ECRYPT Benchmarking of Stream Ciphers. `http://bench.cr.yp.to/results-stream.html` (January 2013)
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak SHA-3 submission. Submission to the NIST SHA-3 Competition (Round 3) (2011), `http://keccak.noekeon.org/Keccak-submission-3.pdf`
9. Biere, A.: PicoSAT Essentials. JSAT 4(2-4), 75–97 (2008)
10. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
11. 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. Springer (2011)
12. Biryukov, A., Nikolić, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert, H. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 322–344. Springer (2010)
13. Bogdanov, A.: Analysis and Design of Block Cipher Constructions. Ph.D. thesis, Ruhr University Bochum (2009)
14. Bogdanov, A.: On unbalanced Feistel networks with contracting MDS diffusion. Des. Codes Cryptography 59(1-3), 35–58 (2011)
15. Bouillaguet, C., Fouque, P.A., Leurent, G.: Security Analysis of SIMD. In: Biryukov et al. [11], pp. 351–368
16. Brummayer, R., Biere, A.: C32SAT: Checking C Expressions. In: Damm and Hermanns [22], pp. 294–297
17. 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)
18. Castro, J.C.H., Estévez-Tapiador, J.M., Quisquater, J.J.: On the Salsa20 Core Function. In: Nyberg [41], pp. 462–469
19. Clavier, C., Gaj, K. (eds.): Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings, Lecture Notes in Computer Science, vol. 5747. Springer (2009)

20. Daemen, J., Govaerts, R., Vandewalle, J.: Resynchronization Weaknesses in Synchronous Stream Ciphers. In: EUROCRYPT. pp. 159–167 (1993)
21. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
22. 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)
23. De Cannière, C.: Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. Lecture Notes in Computer Science, vol. 4176, pp. 171–186. Springer (2006)
24. De Cannière, C., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier and Gaj [19], pp. 272–288
25. De Cannière, C., Preneel, B.: Trivium. In: Robshaw and Billet [44], pp. 244–266
26. 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`
27. Ganesh, V., Dill, D.L.: A Decision Procedure for Bit-Vectors and Arrays. In: Damm and Hermanns [22], pp. 519–531
28. Gueron, S.: Intel's New AES Instructions for Enhanced Performance and Security. In: Dunkelman, O. (ed.) FSE. Lecture Notes in Computer Science, vol. 5665, pp. 51–66. Springer (2009)
29. Josefsson, S., Strombergson, J., Mavrogiannopoulos, N.: The Salsa20 Stream Cipher for Transport Layer Security. IETF Network Working Group (October 2013), `http://tools.ietf.org/html/draft-josefsson-salsa20-tls-02`
30. Jutla, C.S., Patthak, A.C.: A Simple and Provably Good Code for SHA Message Expansion. IACR Cryptology ePrint Archive 2005, 247 (2005)
31. Kanda, M.: Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function. In: Stinson, D.R., Tavares, S.E. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2012, pp. 324–338. Springer (2000)
32. Käsper, E., Schwabe, P.: Faster and Timing-Attack Resistant AES-GCM. In: Clavier and Gaj [19], pp. 1–17
33. 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)
34. 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)
35. 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)
36. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: EUROCRYPT. pp. 81–91 (1992)
37. Mouha, N.: Automated Techniques for Hash Function and Block Cipher Cryptanalysis. Ph.D. thesis, KU Leuven (2012)
38. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Wu, C., Yung, M., Lin, D. (eds.) Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011)
39. Nad, T.: The CodingTool Library. In: Workshop on Tools for Cryptanalysis. pp. 129–130 (2010)

40. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
41. 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)
42. Rivest, R.L.: The MD6 hash function – A proposal to NIST for SHA-3. Submission to the NIST SHA-3 Competition (Round 1) (2008)
43. Rivest, R.L.: OFFICIAL COMMENT: MD6. NIST SHA-3 Mailing List (2009)
44. Robshaw, M.J.B., Billet, O. (eds.): New Stream Cipher Designs - The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986. Springer (2008)
45. Shibutani, K.: On the Diffusion of Generalized Feistel Structures Regarding Differential and Linear Cryptanalysis. In: Biryukov et al. [11], pp. 211–228
46. 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)
47. Soos, M.: CryptoMiniSat2. http://www.msoos.org/cryptominisat2/ (2013)
48. VMware Knowledge Base: VMware View with PCoIP encryption and bandwidth metrics. http://kb.vmware.com/kb/2009122 (March 2013)
49. Wagner, D.: Re-rolled Salsa20 function. http://groups.google.com/group/sci.crypt/msg/0692e3aaf78687a3 (September 2005)
50. 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)
51. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer (2005)
52. 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

A 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 [3, 5]. The other four characteristics can be obtained by flipping the MSB of every XOR word difference.

Each XOR word difference is represented by a pair of parentheses. Inside these parentheses, the bits are given that have an XOR difference of 1. Bit 31 is the MSB. All other bits have XOR difference of 0.

– Characteristic 1:

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

– Characteristic 2:

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

– Characteristic 3:

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

– Characteristic 4:

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