# Observations on the SIMON block cipher family

Stefan Kölbl[1], Gregor Leander[2], and Tyge Tiessen[1]
`stek@dtu.dk, gregor.leander@rub.de, tyti@dtu.dk`

[1] DTU Compute, Technical University of Denmark, Denmark
[2] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany

**Abstract.** In this paper we analyse the general class of functions underlying the Simon block cipher. In particular, we derive efficiently computable and easily implementable expressions for the exact differential and linear behaviour of Simon-like round functions.
Following up on this, we use those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for Simon. Furthermore, we are able to find all characteristics contributing to the probability of a differential for Simon32 and give better estimates for the probability for other variants.
Finally, we investigate a large set of Simon variants using different rotation constants with respect to their resistance against differential and linear cryptanalysis. Interestingly, the default parameters seem to be not always optimal.

**Keywords:** SIMON, differential cryptanalysis, linear cryptanalysis, block cipher, Boolean functions

## 1 Introduction

Lightweight cryptography studies the deployment of cryptographic primitives in resource-constrained environments. This research direction is driven by a demand for cost-effective, small-scale communicating devices such as RFID tags that are a cornerstone in the Internet of Things. Most often the constrained resource is taken to be the chip-area but other performance metrics such as latency [7], code-size [2] and ease of side-channel protection [12]) have been considered as well. Some of these criteria were already treated in Noekeon [9].

The increased importance of lightweight cryptography and its applications has lately been reflected in the NSA publishing two dedicated lightweight cipher families: Simon and Speck [5]. Considering that this is only the third time within four decades that the NSA has published a block cipher, this is quite remarkable. Especially as NIST has started shortly after this publication to investigate the possibilities to standardise lightweight primitives, Simon and Speck certainly deserve a thorough investigation. This is emphasised by the fact that, in contrast to common practice, neither a security analysis nor a justification of the design choices were published by the NSA. This lack of openness necessarily gives rise to curiosity and caution.

In this paper we focus on the SIMON family of block ciphers; an elegant, innovative and very efficient set of block ciphers. There exists already a large variety of papers, mainly focussed on evaluating SIMON's security with regard to linear and differential cryptanalysis. Most of the methods used therein are rather ad-hoc, often only using approximative values for the differential round probability and in particular for the linear square correlation of one round.

**Our Contribution** With this study, we complement the existing work threefold. Firstly we develop an exact closed form expression for the differential probability and a $\log(n)$ algorithm for determining the square correlation over one round. Their accuracy is proven rigorously. Secondly we use these expressions to implement a model of differential and linear characteristics for SAT/SMT solvers which allows us to find the provably best characteristics for different instantiations of SIMON. Furthermore we are able to shed light on how differentials in SIMON profit from the collapse of many differential characteristics. Thirdly by generalising the probability expressions and the SAT/SMT model, we are able to compare the quality of different parameter sets with respect to differential and linear cryptanalysis.

As a basis for our goal to understand both the security of SIMON as well as the choice of its parameter set, we rigorously derive formulas for the differential probabilities and the linear square correlations of the SIMON-like round function that can be evaluated in constant time and time linear in the word size respectively. More precisely, we study differential probabilities and linear correlations of functions of the form

$$S^a(x) \odot S^b(x) + S^c(x)$$

where $S^i(x)$ corresponds to a cyclic left shift of $x$ and $\odot$ denotes the bitwise AND operation.

We achieve this goal by first simplifying this question by considering equivalent descriptions both of the round function as well as the whole cipher (cf. Section 2.4). These simplifications, together with the theory of quadratic boolean functions, result in a clearer analysis of linear and differential properties (cf. Sections 3 and 4). Importantly, the derived simple equations for computing the probabilities of the SIMON round function can be evaluated efficiently and, more importantly maybe, are conceptually very easy. This allows them to be easily used in computer-aided investigations of differential and linear properties over more rounds. It should be noted here that the expression for linear approximations is more complex than the expression for the differential case. However, with respect to the running time of the computer-aided investigations this difference is negligible.

We used this to implement a framework based on SAT/SMT solvers to find the provably best differential and linear characteristics for various instantiations of SIMON (cf. Section 5, in particular Table 1). Furthermore we are able to shed light on how differentials in SIMON profit from the collapse of many differential characteristics by giving exact distributions of the probabilities of these characteristics for chosen differentials. The framework is open source and publicly available to encourage further research [13].

In Section 6 we apply the developed theory and tools to investigate the design space of SIMON-like functions. In particular, using the computer-aided approach, we find that the standard SIMON parameters are not optimal with regard to the best differential and linear characteristics.

As a side result, we improve the probabilities for the best known differentials for several variants and rounds of SIMON. While this might well lead to (slightly) improved attacks, those improved attacks are out of the scope of our work.

Interestingly, at least for SIMON32 our findings indicate that the choices made by the NSA are good but not optimal under our metrics, leaving room for further investigations and questions. To encourage further research, we propose several alternative parameter choices for SIMON32. Here, we are using the parameters that are optimal when restricting the criteria to linear, differential and dependency properties. We encourage further research on those alternative choices to shed more light on the undisclosed design criteria.

We also like to point out that the SIMON key-scheduling was not part of our investigations. Its influence on the security of SIMON is left as an important open question for further investigations. In line with this, whenever we investigate multi-round properties of SIMON in our work, we implicitly assume independent round keys in the computation of probabilities.

Finally, we note that most of our results can be applied to more general constructions, where the involved operations are restricted to AND, XOR, and rotations.

**Related Work** There are various papers published on the cryptanalysis of SIMON [1,3,6,17,18,19]. The most promising attacks so far are based on differential and linear cryptanalysis, however a clear methodology of how to derive the differential probabilities and square correlations seems to miss in most cases. Biryukov, Roy and Velichkov [6] derive a correct, but rather involved method to find the differential probabilities. Abed, List, Lucks and Wenzel [1] state an algorithm for the calculation of the differential probabilities but without further explanation. For the calculation of the square correlations an algorithm seems to be missing all together.

Previous work also identifies various properties like the strong differential effect and give estimate of the probability of differentials.

The concept behind our framework was previously also applied on the ARX cipher Salsa20 [14] and the CAESAR candidate NORX [4]. In addition to the applications proposed in previous work we extend it for linear cryptanalysis, examine the influence of rotation constants and use it to compute the distribution of characteristics corresponding to a differential.

## 2 Preliminaries

In this section, we start by defining our notation and giving a short description of the round function. We recall suitable notions of equivalence of Boolean functions that allow us to simplify our investigations of SIMON-like round functions. Most of this section is generally applicable to AND-RX constructions, i.e. constructions that only make use of the bitwise operations AND, XOR, and rotations.

## 2.1 Notation

We denote by $\mathbb{F}_2$ the field with two elements and by $\mathbb{F}_2^n$ the $n$-dimensional vector space over $\mathbb{F}_2$. By $\mathbf{0}$ and $\mathbf{1}$ we denote the vectors of $\mathbb{F}_2^n$ with all 0s and all 1s respectively. The Hamming weight of a vector $a \in \mathbb{F}_2^n$ is denoted as $\mathrm{wt}(a)$. By $\mathbb{Z}_n$ we denote the integers modulo $n$.

The addition in $\mathbb{F}_2^n$, i.e. bit-wise XOR, is denoted by $+$. By $\odot$ we denote the AND operation in $\mathbb{F}_2^n$, i.e. multiplication over $\mathbb{F}_2$ in each coordinate:

$$x \odot y = (x_i y_i)_i.$$

By $\vee$ we denote the bitwise OR operation. By $\overline{x}$ we denote the bitwise negation of $x$, i.e. $\overline{x} := (x + \mathbf{1})$. We denote by $S^i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ the left circular shift by $i$ positions. We also note that any arithmetic of bit indices is always done modulo the word size $n$.

In this paper we are mainly concerned with functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x) \tag{1}$$

and we identify such functions with its triple $(a, b, c)$ of parameters.

For a vectorial Boolean function on $n$ bits, $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we denote by

$$\widehat{f}(\alpha, \beta) = \sum_x \mu\left(\langle \beta, f \rangle + \langle \alpha, x \rangle\right)$$

the Walsh (or Fourier) coefficient with input mask $\alpha$ and output mask $\beta$. Here we use $\mu(x) = (-1)^x$ to simplify the notation.

The corresponding squared correlation of $f$ is given by

$$C^2(\alpha \to \beta) = \left(\frac{\widehat{f}(\alpha, \beta)}{2^n}\right)^2.$$

For differentials we similarly denote by $\Pr(\alpha \longrightarrow \beta)$ the probability that a given input difference $\alpha$ results in a given output difference $\beta$, i.e.

$$\Pr(\alpha \longrightarrow \beta) = \frac{|\{x \mid f(x) + f(x + \alpha) = \beta\}|}{2^n}.$$

Furthermore, $\mathsf{Dom}(f)$ is the domain of a function $f$, $\mathsf{Img}(f)$ is its image.

## 2.2 Description of SIMON

Simon is a family of lightweight block ciphers with block sizes 32, 48, 64, 96, and 128 bits. The constructions are Feistel ciphers using a word size $n$ of 16, 24, 32, 48 or 64 bits, respectively. We will denote the variants as Simon$2n$. The key size varies between of 2, 3, and 4 $n$-bit words. The round function of Simon is composed of AND, rotation, and XOR operations on the complete word (see figure 1). More precisely, the round function in Simon corresponds to

$$S^8(x) \odot S^1(x) + S^2(x),$$

that is to the parameters $(8, 1, 2)$ for $f$ as given in Equation (1). As we are not only interested in the original Simon parameters, but in investigating the entire design space of Simon-like functions, we denote by

$$\textsc{Simon}[a, b, c]$$

the variant of Simon where the original round function is replaced by $f_{a,b,c}$ (cf. Equation (1)).

As it is out of scope for our purpose, we refer to [5] for the description of the key-scheduling.

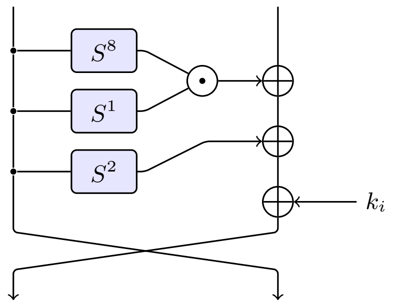

**Fig. 1.** The round function of Simon.

## 2.3 Affine equivalence of Boolean Functions

Given two (vectorial) Boolean functions $f_1$ and $f_2$ on $\mathbb{F}_2^n$ related by

$$f_1(x) = (A \circ f_2 \circ B)(x) + C(x)$$

where $A$ and $B$ are affine permutations and $C$ is an arbitrary affine mapping on $\mathbb{F}_2^n$ we say that $f_1$ and $f_2$ are *extended affine equivalent* (cf. [8] for a comprehensive survey).

With respect to differential cryptanalysis, if $f_1$ and $f_2$ are extended affine equivalent then the differential $\alpha \xrightarrow{f_1} \beta$ over $f_1$ has probability $p$ if and only if the differential

$$B(\alpha) \xrightarrow{f_2} A^{-1}(\beta + C(\alpha))$$

over $f_2$ has probability $p$ as well.

For linear cryptanalysis, a similar relation holds for the linear correlation. If $f_1$ and $f_2$ are related as defined above, it holds that

$$\widehat{f_1}(\alpha, \beta) = \widehat{f_2}\left(\left(C \circ B^{-1}\right)^T \beta + \left(B^{-1}\right)^T \alpha, A^T \beta\right).$$

Thus up to linear changes we can study $f_2$ instead of $f_1$ directly. Note that, for an actual attack, these changes are usually critical and can certainly not be ignored. However, tracing the changes is, again, simple linear algebra.

For differential and linear properties of Simon-like functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x)$$

this implies that it is sufficient to find the differential and linear properties of the simplified variant

$$f(x) = x \odot S^d(x)$$

and then transfer the results back by simply using linear algebra.[3]

## 2.4 Structural Equivalence Classes in AND-RX Constructions

AND-RX constructions, i.e. constructions that make only use of the operations AND ($\odot$), XOR ($+$), and rotations ($S^r$), exhibit a high degree of symmetry. Not only are they invariant under rotation of all input words, output words and constants, they are furthermore structurally invariant under any affine transformation of the bit-indices. As a consequence of this, several equivalent representations of the Simon variants exist.

Let $T$ be a permutation of the bits of an $n$-bit word that corresponds to an affine transformation of the bit-indices. Thus there are $s \in \mathbb{Z}_n^*$ and $t \in \mathbb{Z}_n$ such that bit $i$ is renamed to $s \cdot i + t$. As the AND and XOR operations are bitwise, $T$ clearly commutes with these:

$$\begin{aligned} Tv \odot Tw &= T(v \odot w) \\ Tv + Tw &= T(v + w) \end{aligned}$$

where $v$ and $w$ are $n$-bit words. A rotation to the left by $r$ can be written bitwise as $S^r(v)_i = v_{i-r}$. For a rotation, we thus get the following bitwise relation after transformation with $T$

$$S^r(v)_{s \cdot i + t} = v_{s \cdot (i-r)+t} = v_{s \cdot i + t - s \cdot r} \quad .$$

Substituting $s \cdot i + t$ with $j$ this is the same as

$$S^r(v)_j = v_{j - s \cdot r} \quad .$$

[3] Note that we can transform the equation $f(x) = S^a(x) \odot S_b(x) + S^c(x)$ to the equation $S^{-a}(f(x)) + S^{c-a}(x)) = x \odot S^{b-a}(x)$.

Thus the rotation by $r$ has been changed to a rotation by $s \cdot r$. Thus we can write

$$TS^r v = S^{s \cdot r} T v.$$

Commuting the linear transformation of the bit-indices with a rotation thus only changes the rotation constant by a factor. In the special case where all input words, output words and constants are rotated, which corresponds to the case $s = 1$, the rotation constant are left untouched.

To summarise the above, when applying such a transformation $T$ to all input words, output words and constants in an AND-RX construction, the structure of the constructions remains untouched apart from a multiplication of the rotation constants by the factor $s$.

This means for example for SIMON32 that changing the rotation constants from $(8, 1, 2)$ to $(3 \cdot 8, 3 \cdot 1, 3 \cdot 2) = (8, 3, 6)$ and adapting the key schedule accordingly gives us the same cipher apart from a bit permutation. As $s$ has to be coprime to $n$, all $s$ with $\gcd(s, n) = 1$ are allowed, giving $\varphi(n)$ equivalent tuples of rotation constants in each equivalence class where $\varphi$ is Euler's phi function.

Together with the result from section 2.3, this implies the following lemma.

**Lemma 1.** *Any function $f_{a,b,c}$ as defined in Equation (1) is extended affine equivalent to a function*

$$f(x) = x \odot S^d(x)$$

*where $d|n$ or $d = 0$ .*

When looking at differential and square correlations of SIMON-like round functions this means that it is sufficient to investigate this restricted set of functions. The results for these functions can then simply be transferred to the general case.

## 3 Differential Probabilities of SIMON-like round functions

In this section, we derive a closed expression for the differential probability for all SIMON-like round functions, i.e. all functions as described in Equation (1). The main ingredients here are the derived equivalences and the observation that any such function is quadratic. Being quadratic immediately implies that its derivative is linear and thus the computation of differential probabilities basically boils down to linear algebra (cf. Theorem 1). However, to be able to efficiently study multiple-round properties and in particular differential characteristics, it is important to have a simple expression for the differential probabilities. Those expressions are given for $f(x) = x \odot S^1(x)$ in Theorem 2 and for the general case in Theorem 3.

### 3.1 A closed expression for the differential probability

The following statement summarises the differential properties of the $f$ function.

**Theorem 1.** *Given an input difference $\alpha$ and an output difference $\beta$ the probability $p$ of the corresponding differential (characteristic) for the function $f(x) = x \odot S^a(x)$ is given by*

$$p_{\alpha,\beta} = \begin{cases} 2^{-(n-d)} & \text{if } \beta + \alpha \odot S^a(\alpha) \in \mathsf{Img}(L_\alpha) \\ 0 & \text{else} \end{cases}$$

*where*

$$d = \dim \ker(L_\alpha)$$

*and*

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x)$$

*Proof.* We have to count the number of solutions to the equation

$$f(x) + f(x + \alpha) = \beta.$$

This simplifies to

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x) = \beta + \alpha \odot S^a(\alpha)$$

As this is an affine equation, it either has zero solutions or the number of solutions equals the kernel size, i.e. the number of elements in the subspace

$$\{x \mid x \odot S^a(\alpha) + \alpha \odot S^a(x) = \mathbf{0}\}.$$

Clearly, the equation has solutions if and only if $\beta + \alpha \odot S^a(\alpha)$ is in the image of $L_\alpha$. □

Next we present a closed formula to calculate the differential probability in the case where $a = 1$. Furthermore we restrict ourselves to the case where $n$ is even.

**Theorem 2.** *Let*

$$\texttt{varibits} = S^1(\alpha) \vee \alpha$$

*and*

$$\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha).$$

*Then the probability that difference $\alpha$ goes to difference $\beta$ is*

$$P(\alpha \to \beta) = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \mathrm{wt}(\beta) \equiv 0 \mod 2 \\ 2^{-\mathrm{wt}(\texttt{varibits}+\texttt{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \beta \odot \overline{\texttt{varibits}} = \mathbf{0} \\ & \text{and } (\beta + S^1(\beta)) \odot \texttt{doublebits} = \mathbf{0} \\ 0 & \text{else} \end{cases}$$

*Proof.* According to theorem 1, we need to prove two things. Firstly we need to prove that the rank of $L_\alpha$ (i.e. $n - \dim\ker L_\alpha$) is $n-1$ when $\alpha = \mathbf{1}$, and $\mathrm{wt}(\texttt{varibits} + \texttt{doublebits})$) otherwise. Secondly we need to prove that $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$ iff $\mathrm{wt}(\beta) \equiv 0 \mod 2$ when $\alpha = \mathbf{1}$, and that $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$ iff $\beta \odot \texttt{varibits} = \mathbf{0}$ and $(\beta + S^1(\beta)) \odot \texttt{doublebits} = \mathbf{0}$ when $\alpha \neq \mathbf{1}$.

We first consider the first part. Let us write $L_\alpha(x)$ in matrix form and let us take $x$ to be a column vector. $S^1(\alpha) \odot x$ can be written as $M_{S^1(\alpha)\odot} x$ with

$$M_{S^1(\alpha)\odot} = \begin{pmatrix} \alpha_{n-1} & \dots\dots & 0 \\ \vdots & \alpha_0 & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \dots\dots & \alpha_{n-2} \end{pmatrix} . \tag{2}$$

Equivalently we can write $\alpha \odot x$ and $S^1(x)$ with matrices as $M_{\alpha\odot} x$ and $M_{S^1} x$ respectively where

$$M_{\alpha\odot} = \begin{pmatrix} \alpha_0 & \dots\dots & 0 \\ \vdots & \alpha_1 & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \dots\dots & \alpha_{n-1} \end{pmatrix} \text{ and } M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix} , \tag{3}$$

i.e. $M_{S^1}$ consists of two identity and two zero submatrices. The result of $M_{S^1(\alpha)\odot} + M_{\alpha\odot} M_{S^1}$ can now be written as

$$\begin{pmatrix} \alpha_{n-1} & 0 & 0 & \dots & \alpha_0 \\ \alpha_1 & \alpha_0 & 0 & \dots & 0 \\ 0 & \alpha_2 & \alpha_1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \alpha_{n-1} & \alpha_{n-2} \end{pmatrix} \tag{4}$$

Clearly the rank of the matrix is $n-1$ when all $\alpha_i$ are 1. Suppose now that not all $\alpha_i$ are 1. In that case, a set of non-zero rows is linearly dependent iff there exist two identical rows in the set. Thus to calculate the rank of the matrix, we need to calculate the number of unique non-zero rows.

By associating the rows in the above matrix with the bits in `varibits`, we can clearly see that the number of non-zero rows in the matrices corresponds to the number of 1s in $\texttt{varibits} = S^1(\alpha) \vee \alpha$.

To count the number of non-unique rows, first notice that a nonzero row can only be identical to the row exactly above or below. Suppose now that a non-zero row $i$ is identical to the row $(i-1)$ above. Then $\alpha_{i-1}$ has to be 0 while $\alpha_i$ and $\alpha_{i-2}$ have to be 1. But then row $i$ cannot simultaneously be identical to row $(i+1)$ below. Thus it is sufficient to calculate the number of non-zero rows minus the number of rows that are identical to the row above it to find the rank of the matrix. Noting that row $i$ is non-zero iff $\alpha_i \alpha_{i-1}$ and that $\alpha_i \overline{\alpha_{i-1}} \alpha_{i-2}$ is only equal 1 when row $i$ is non-zero and equal to the row above it. Thus calculating the number of $i$ for which

$$\alpha_i \alpha_{i-1} + \alpha_i \overline{\alpha_{i-1}} \alpha_{i-2}$$

is equal 1 gives us the rank of $L_\alpha$. This corresponds to calculating $\mathrm{wt}(\mathtt{varibits} + \mathtt{doublebits})$.

For the second part of the proof, we need to prove the conditions that check whether $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$. First notice that $\alpha \odot S^1(\alpha)$ is in the image of $L_\alpha$ (consider for $x$ the vector with bits alternately set to 0 and 1). Thus it is sufficient to test whether $\beta$ is in $\mathsf{Img} L_\alpha$. Let $y = L_\alpha(x)$. Let us first look at the the case of $\alpha = \mathbf{1}$. Then $L_\alpha(x) = x + S^1(x)$. We can thus deduce from bit $y_i$ whether $x_i = x_{i-1}$ or $x_i \neq x_{i-1}$. Thus the bits in $y$ create a chain of equalities/inequalities in the bits of $x$ which can only be fulfilled if there the number of inequalities is even. Hence in that case $\beta \in \mathsf{Img} L_\alpha$ iff $\mathrm{wt}(\beta) \equiv 0 \mod 2$.

For the case that $\alpha \neq \mathbf{1}$, we first note that $y_i$ has to be zero if the corresponding row $i$ in the matrix of equation (4) is all zeroes. Furthermore following our discussion of this matrix earlier, we see that $y_i$ is independent of the rest of $y$ if the corresponding row is linearly independent of the other rows and that $y_i$ has to be the same as $y_{i-1}$ if the corresponding rows are identical. Thus we only need to check that the zero-rows of the matrix correspond to zero bits in $\beta$ and that the bits in $\beta$ which correspond to identical rows in the matrix are equal. Thus $\beta$ is in the image of $L_\alpha$ iff $\beta \odot \overline{\mathtt{varibits}} = \mathbf{0}$ and $(\beta + S^1(\beta)) \odot \mathtt{doublebits} = \mathbf{0}$. □

### 3.2 The full formula for differentials.

Above we treated only the case for the simplified function $f(x) = x \cdot S^1(x)$. As mentioned earlier, the general case where $\gcd(a-b, n) = 1$ can be deduced from this with linear algebra. When $\gcd(d, n) \neq 1$ though, the function $f(x) = x \odot S^d(x)$ partitions the output bits into independent classes. This not only raises differential probabilities (worst case $d = 0$), it also makes the notation for the formulas more complex and cumbersome, though not difficult. We thus restrict ourselves to the most important case when $\gcd(a-b, n) = 1$. The general formulas are then

**Theorem 3.** *Let $f(x) = S^a(x) \odot S^b(x) + S^c(x)$, where $\gcd(n, a-b) = 1$, $n$ even, and $a > b$ and let $\alpha$ and $\beta$ be an input and an output difference. Then with*

$$\mathtt{varibits} = S^a(\alpha) \vee S^b(\alpha)$$

*and*

$$\mathtt{doublebits} = S^b(\alpha) \odot \overline{S^a(\alpha)} \odot S^{2a-b}(\alpha)$$

*and*

$$\gamma = \beta + S^c(\alpha)$$

*we have that the probability that difference $\alpha$ goes to difference $\beta$ is*

$$P(\alpha \to \beta) = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \operatorname{wt}(\gamma) \equiv 0 \mod 2 \\ 2^{-\operatorname{wt}(\texttt{varibits}+\texttt{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \gamma \odot \overline{\texttt{varibits}} = \mathbf{0} \\ & \text{and } (\gamma + S^{a-b}(\gamma)) \odot \texttt{doublebits} = \mathbf{0} \\ 0 & \text{else} \ . \end{cases}$$

For a more intuitive approach and some elaboration on the differential probabilities, we refer to the ePrint version of this paper.

## 4 Linear Correlations of SIMON-like round functions

As in the differential case, for the study of linear approximations, we also build up on the results from subsections 2.3 and 2.4. We will thus start with studying linear approximations for the function $f(x) = x \odot S^a(x)$. Again, the key point here is that all those functions are quadratic and thus their Fourier coefficient, or equivalently their correlation, can be computed by linear algebra (cf. Theorem 4). Theorem 5 is then, in analogy to the differential case, the explicit expression for the linear correlations. It basically corresponds to an explicit formula for the dimension of the involved subspace.

The first result is the following:

**Theorem 4.**

$$\widehat{f}(\alpha,\beta)^2 = \begin{cases} 2^{n+d} & \text{if } \alpha \in U_\beta^\perp \\ 0 & \text{else} \end{cases}$$

*where*

$$d = \dim U_\beta$$

*and*

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

*Proof.* We compute

$$\begin{aligned} \widehat{f}(\alpha,\beta)^2 &= \sum_{x,y} \mu\left(\langle \beta, f(x) + f(y)\rangle + \langle \alpha, x+y\rangle\right) \\ &= \sum_{x,y} \mu\left(\langle \beta, f(x) + f(x+y)\rangle + \langle \alpha, y\rangle\right) \\ &= \sum_{x,y} \mu\left(\langle \beta, x \odot S^a(x) + (x+y) \odot S^a(x+y)\rangle + \langle \alpha, y\rangle\right) \\ &= \sum_{y} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right) \sum_{x} \mu\left(\langle \beta, x \odot S^a(y) + y \odot S^a(x)\rangle\right) \\ &= \sum_{y} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right) \sum_{x} \mu\left(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y)\rangle\right) \ . \end{aligned}$$

Now for the sum over $x$ only two outcomes are possible, $2^n$ or zero. More precisely, it holds that

$$\sum_x \mu\left(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y)\rangle\right) = \begin{cases} 2^n & \text{if } \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0} \\ 0 & \text{else} \quad . \end{cases}$$

Thus, defining

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

we get

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right).$$

Now as

$$\begin{aligned} \langle \beta, f(y)\rangle =& \langle \beta, y \odot S^a(y)\rangle && (5)\\ =& \langle \mathbf{1}, y \odot \beta \odot S^a(y)\rangle && (6)\\ =& \langle \mathbf{1}, y \odot S^{-a}(\beta \odot y)\rangle && (7)\\ & && (8) \end{aligned}$$

Now, the function $f_\beta := \langle \beta, f(y)\rangle$ is linear over $U_\beta$ as can be easily seen by the definition of $U_\beta$. Moreover, as $f_\beta$ is unbalanced for all $\beta$, it follows that actually $f_\beta$ is constant zero on $U_\beta$. We thus conclude that

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu\left(\langle \alpha, y\rangle\right).$$

With a similar argument as above, it follows that $\widehat{f}(\alpha, \beta)^2$ is non-zero if and only if $\alpha$ is contained in $U_\beta^\perp$. □

Let us now restrict ourselves to the case where $f(x) = x \odot S^1(x)$. The general case can be deduced analogously to the differential probabilities. For simplicity we also restrict ourselves to the case where $n$ is even.

First we need to introduce some notation. Let $x \in \mathbb{F}_2^n$ with not all bits equal to 1. We now look at blocks of consecutive 1s in $x$, including potentially a block that "wraps around" the ends of $x$. Let the lengths of these blocks, measured in bits, be denoted as $c_0, \ldots, c_m$. For example, the bitstring 100101111011 has blocks of length 1, 3, and 4. With this notation define $\theta(x) := \sum_{i=0}^{m} \lceil \frac{c_i}{2} \rceil$.

Noting that the linear square correlation of $f$ is $\frac{\widehat{f}(\alpha,\beta)^2}{2^{2n}}$, we then have the following theorem:

**Theorem 5.** *With the notation from above it holds that the linear square correlation of $\alpha \xrightarrow{f} \beta$ can be calculated as*

$$C(\alpha \to \beta) = \begin{cases} 2^{-n+2} & \text{if } \beta = \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 2^{-\theta(\beta))} & \text{if } \beta \neq \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 0 & \text{else.} \end{cases}$$

*Proof.* Define $L_\beta(x) := \beta \odot S^1(x) + S^{-1}(\beta \odot x)$. Clearly $L_\beta$ is linear. Also $U_\beta = \ker L_\beta(x)$. Let us determine the rank of this mapping. Define the matrices $M_{\beta\cdot}$, $M_{S^1}$, and $M_{S^{-1}}$ as

$$M_{\beta\cdot} = \begin{pmatrix} \beta_0 & \dots\dots & & 0 \\ \vdots & \beta_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots\dots & & \beta_{n-1} \end{pmatrix} \qquad \begin{matrix} M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix} \\ \\ M_{S^{-1}} = \begin{pmatrix} 0_{n-1,1} & I_{n-1,n-1} \\ I_{1,1} & 0_{1,n-1} \end{pmatrix} \end{matrix} \tag{9}$$

We can then write $L_\beta$ in matrix form as

$$\begin{pmatrix} 0 & \beta_1 & 0 & \dots & 0 & \beta_0 \\ \beta_1 & 0 & \beta_2 & 0 & \dots & 0 \\ 0 & \beta_2 & 0 & \beta_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \beta_{n-1} \\ \beta_0 & 0 & \dots & 0 & \beta_{n-1} & 0 \end{pmatrix} \tag{10}$$

Clearly, if $\beta$ is all 1s, the rank of the matrix is $n-2$ as $n$ is even.[4] Let us therefore now assume that $\beta$ is not all 1s. When we look at a block of 1s in $\beta$ e.g., $\beta_{i-1} = 0$, $\beta_i, \beta_{i+1}, \dots, \beta_{i+l-1} = 1$, and $\beta_l = 0$. Then clearly the $l$ rows $i, i+1, \dots, i+l-1$ are linearly independent when $l$ is even. When $l$ is odd though, the sum of rows $i$, $i+2$, $i+4$, up to row $i+l-3$ will equal row $i+l-1$. In that case there are thus only $l-1$ linearly independent rows. As the blocks of 1s in $\beta$ generate independent blocks of rows, we can summarise that the rank of the matrix is exactly $\theta(\beta)$. □

Analogously to the differential probabilities, the linear probabilities in the general case can be derived from this. It is likewise straightforward to derive how to determine whether $\alpha \in U_\beta^\perp$. As an explicit formulation of this is rather tedious, we instead refer to the implementation in Python given in the Appendix B where both is achieved in the case where $\gcd(a-b, n) = 1$ and $n$ is even.

For a more intuitive approach and some elaboration on the linear probabilities, we refer to the ePrint version of this paper.

## 5 Finding Optimal Differential and Linear Characteristics

While there are various methods for finding good characteristics, determining optimal differential or linear characteristics remains a hard problem in general. The formulas derived for both differential and linear probabilities enable us to apply an algebraic approach to finding the best characteristics. A similar technique has been applied to the ARX cipher Salsa20 [14] and the CAESAR candidate

[4] The rank is $n-1$ when $n$ is odd.

NORX [4]. For finding the optimal characteristics for Simon we implemented an open source tool [13] based on the SAT/SMT solvers CryptoMiniSat [15] and STP [11].

In the next section we will show how Simon can be modelled to find both the best differential and linear characteristics in this framework and how this can be used to solve cryptanalytic problems.

### 5.1 Model for Differential Cryptanalysis of SIMON

First we define the variables used in the model of Simon. We use two $n$-bit variables $x_i$, $y_i$ to represent the XOR-difference in the left and right halves of the state for each round and an additional variable $z_i$ to store the XOR-difference of the output of the AND operation.

For representing the $\log_2$ probability of the characteristic we introduce an additional variable $w_i$ for each round. The sum over all probabilities $w_i$ then gives the probability of the corresponding differential characteristic. The values $w_i$ are computed according to theorem 3 as

$$w_i = \mathrm{wt}(\texttt{varibits} + \texttt{doublebits}) \tag{11}$$

where $\mathrm{wt}(x)$ is the Hamming weight of $x$ and

$$\begin{aligned} \texttt{varibits} &= S^a(x_i) \vee S^b(x_i) \\ \texttt{doublebits} &= S^b(x_i) \odot \overline{S^a(x_i)} \wedge S^{2a-b}(x_i) \end{aligned}$$

Therefore, for one round of Simon we get the following set of constraints:

$$\begin{aligned} y_{i+1} &= x_i \\ 0 &= (z_i \odot \texttt{varibits}) \\ 0 &= (z_i + S^{a-b}(z_i)) \odot \texttt{doublebits} \\ x_{i+1} &= y_i + z_i + S^c(x_i) \\ w_i &= \mathrm{wt}(\texttt{varibits} + \texttt{doublebits}) \end{aligned} \tag{12}$$

A model for linear characteristics, though slightly more complex, can be implemented in a similar way. A description of this model can be found in the implementation of our framework. Despite the increase in complexity, we could not observe any significant impact on the solving time for the linear model.

### 5.2 Finding Optimal Characteristics

We can now use the previous model for Simon to search for optimal differential characteristics. This is done by formulating the problem of finding a valid characteristic, with respect to our constraints, for a given probability $w$. This is important to limit the search space and makes sense as we are usually more interested in differential characteristics with a high probability as they are more promising to lead to attacks with a lower complexity. Therefore, we start with a high probability and check if such a characteristic exists. If not we lower the probability.

The procedure can be described in the following way:

– For each round of the cipher add the corresponding constraints as defined in (12). This system of constraints then exactly describes the form of a valid characteristic for the given parameters.
– Add a condition which accumulates the probabilities of each round as defined in (11) and check if it is equal to our target probability $w$.
– Query if there exists an assignment of variables which is satisfiable under the constraints.
– Decrement the probability $w$ and repeat the procedure.

One of the main advantages compared to other approaches is that we can prove an upper bound on the probability of characteristics for a given cipher and number of rounds. If the solvers determines the set of conditions unsatisfiable, we know that no characteristic with the specified probability exists. We used this approach to determine the characteristics with optimal probability for different variants of SIMON. The results are given in Table 1.

**Table 1.** Overview of the optimal differential (on top) and linear characteristics for different variants of SIMON. The probabilities are given as $\log_2(p)$, for linear characteristic the squared correlation is used.

| Rounds: | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Differential** | | | | | | | | | | | | | | | |
| SIMON32 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −25 | −30 | −34 | −36 | −38 | −40 | −42 |
| SIMON48 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −35 | −38 | −44 | −46 | −50 |
| SIMON64 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −48 | −54 |
| **Linear** | | | | | | | | | | | | | | | |
| SIMON32 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −34 | −36 | −38 | −40 | −42 |
| SIMON48 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −46 | −50 |
| SIMON64 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −48 | −54 |

**Upper Bound for the Characteristics.** During our experiments we observed that it seems to be an easy problem for the SMT/SAT solver to prove the absence of differential characteristics above $w_{\max}$. This can be used to get a lower bound on the probability of characteristics contributing to the differential. The procedure is similar to finding the optimal characteristics.

– Start with a very low initial probability $w_i$.
– Add the same system of constraints which were used for finding the characteristic.

– Add a constraint fixing the variables $(x_0, y_0)$ to $\Delta_{\text{in}}$ and $(x_r, y_r)$ to $\Delta_{\text{out}}$.
– Query if there is a solution for this weight.
– Increase the probability $w_i$ and repeat the procedure until a solution is found.

### 5.3 Computing the Probability of a Differential

Given a differential characteristic it is of interest to determine the probability of the associated differential $\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}})$ as it might potentially have a much higher probability then the single characteristic. It is often assumed that the probability of the best characteristic can be used to approximate the probability of the best differential. However, this assumption only gives an inaccurate estimate in the case of Simon.

Similarly to the previous approach for finding the characteristic, we can formalise the problem of finding the probability of a given differential in the following way:

– Add the same system of constraints which were used for finding the characteristic.
– Add a constraint fixing the variables $(x_0, y_0)$ to $\Delta_{\text{in}}$ and $(x_r, y_r)$ to $\Delta_{\text{out}}$.
– Use a SAT solver to find **all** solutions $s_i$ for the probability $w$.
– Decrement the probability $w$ and repeat the procedure.

The probability of the differential is then given by

$$\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}}) = \sum_{i=w_{\min}}^{w_{\max}} s_i \cdot 2^{-i} \tag{13}$$

where $s_i$ is the number of characteristics with a probability of $2^{-i}$.

We used this approach to compute better estimates for the probability of various differentials (see Table 2). In the case of Simon32 we were able to find *all* characteristics contributing to the differentials for 13 and 14 rounds. The distribution of the characteristics and accumulated probability of the differential is given in Figure 2. It is interesting to see that the distribution of $w$ in the range $[55, 89]$ is close to uniform and therefore the probability of the corresponding differential improves only negligible and converges quickly towards the measured probability[5].

The performance of the whole process is very competitive compared to dedicated approaches. Enumerating all characteristics up to probability $2^{-46}$ for the 13-round Simon32 differential takes around 90 seconds on a single CPU core and already gives a better estimate compared to the results in [6]. A complete enumeration of all characteristics for 13-round Simon32 took close to one core month using CryptoMiniSat4 [15]. The computational effort for other variants of Simon is comparable given the same number of rounds. However, for these

[5] We encrypted all $2^{32}$ possible texts under 100 random keys to obtain the estimate of the probability for 13-round Simon32.

variants we can use differentials with a lower probability covering more rounds due to the increased block size. In this case the running time increases due to the larger interval $[w_{\min}, w_{\max}]$ and higher number of rounds.

For Simon48 and Simon64 we are able to improve the estimate given in [16]. Additionally we found differentials which can cover 17 rounds for Simon48 and 22 rounds for Simon64 which might have potential to improve previous attacks. Our results are also closer to the experimentally obtained estimates given in [10] but give a slightly lower probability. This can be due to the limited number of characteristics we use for the larger Simon variants or the different assumptions on the independence of rounds.

Our results are limited by the available computing power and in general it seems to be difficult to count all characteristics for weights in $[w_{\min}, w_{\max}]$, especially for the larger variants of Simon. However the whole process is embarrassingly parallel, as one can split up the computation for each probability $w_i$. Furthermore, the improvement that one gets decreases quickly. For all differentials we observed that the distribution of differential characteristics becomes flat after a certain point.

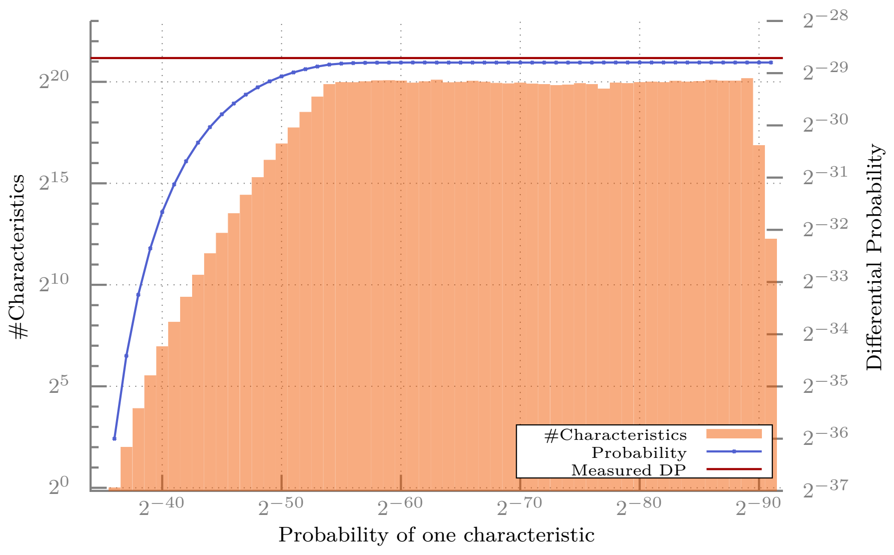

**Fig. 2.** Distribution of the number of characteristics for the differential $(0, 40) \rightarrow (4000, 0)$ for 13-round Simon32 and the accumulated probability. A total of $\approx 2^{25.21}$ characteristics contribute to the probability.

**Table 2.** Overview of the differentials and the range $[w_{\min}, w_{\max}]$ of the $\log_2$ probabilities of the characteristics contributing to the differential. For computing the lower bound $\log_2(p)$ of the probability of the differentials, we used all characteristics with probabilities in the range from $w_{\min}$ up to the values in brackets in the $w_{\max}$ column.

| Cipher | Rounds | $\Delta_{\text{in}}$ | $\Delta_{\text{out}}$ | $w_{\min}$ | $w_{\max}$ | $\log_2(p)$ |
|---|---|---|---|---|---|---|
| Simon32 | 13 | $(0, 40)$ | $(4000, 0)$ | 36 | 91 (91) | $-28.79$ |
| Simon32 | 14 | $(0, 8)$ | $(800, 0)$ | 38 | 120 (120) | $-30.81$ |
| Simon48 | 15 | $(20, 800088)$ | $(800208, 2)$ | 46 | 219 (79) | $-41.02$ |
| Simon48 | 16 | $(800000, 220082)$ | $(800000, 220000)$ | 50 | 256 (68) | $-44.33$ |
| Simon48 | 17 | $(80, 222)$ | $(222, 80)$ | 52 | 269 (85) | $-46.32$ |
| Simon64 | 21 | $(4000000, 11000000)$ | $(11000000, 4000000)$ | 68 | 453 (89) | $-57.57$ |
| Simon64 | 22 | $(440, 1880)$ | $(440, 100)$ | 72 | 502 (106) | $-61.32$ |

## 6 Analysis of the Parameter Choices

The designers of Simon so far gave no justification for their choice of the rotation constants. Here we evaluate the space of rotation parameters with regard to different metrics for the quality of the parameters. Our results are certainly not a definite answer but are rather intended as a starting point to evaluating the design space and reverse engineering the design choices. We consider all possible sets of rotation constants $(a, b, c)$[6] and checked them for diffusion properties and the optimal differential and linear characteristics.

### 6.1 Diffusion

As a very simple measure to estimate the quality of the rotation constants, we measure the number of rounds that are needed to reach full diffusion. Full diffusion is reached when every state bit principally depends on all input bits. Compared to computing linear and differential properties it is an easy task to determine the dependency.

In Table 3 we give a comparison to how well the standard Simon rotation parameters fare within the distribution of all possible parameter sets. The exact distributions for all Simon variants can be found in the appendix in Table 8.

### 6.2 Differential and Linear

As a second criteria for our parameters, we computed for all $a > b$ and $\gcd(a - b, n) = 1$ the optimal differential and linear characteristics for 10 rounds of Simon32, Simon48 and Simon64. A list of the parameters which are optimal for all three variants of Simon can be found in Appendix D.

It is important here to note that there are also many parameter sets, including the standard choice, for which the best 10-round characteristics of Simon32 have

[6] Without lack of generality, we assume though that $a \geq b$.

**Table 3.** The number of rounds after which full diffusion is reached for the standard SIMON parameters in comparison to the whole possible set of parameters.

| Block size | 32 | 48 | 64 | 96 | 128 |
|---|---|---|---|---|---|
| Standard parameters | 7 | 8 | 9 | 11 | 13 |
| Median | 8 | 10 | 11 | 13 | 14 |
| First quartile | 7 | 9 | 9 | 11 | 12 |
| Best possible | 6 | 7 | 8 | 9 | 10 |
| Rank | 2nd | 2nd | 2nd | 3rd | 4th |

a probability of $2^{-25}$ compared to the optimum of $2^{-26}$. However, this difference by a factor of 2 does not seem to occur for more than 10 rounds and also not any larger variants of SIMON.

### 6.3 Interesting Alternative Parameter Sets

As one result of our investigation we chose three exemplary sets of parameters that surpass the standard parameters with regards to some metrics. Those variants are SIMON$[12, 5, 3]$, SIMON$[7, 0, 2]$ and SIMON$[1, 0, 2]$.

SIMON$[12, 5, 3]$ has the best diffusion amongst the parameters which have optimal differential and linear characteristics for 10 rounds. The two other choices are both restricted by setting $b = 0$ as this would allow a more efficient implementation in software. Among those SIMON$[7, 0, 2]$ has the best diffusion and the characteristics behave similar to the standard parameters. Ignoring the diffusion SIMON$[1, 0, 2]$ seems also an interesting choice as it is optimal for the differential and linear characteristics.

If we look though at the differential corresponding to the best differential characteristic of SIMON$[7, 0, 2]$ and SIMON$[1, 0, 2]$, then we can see the number of characteristics contributing to it is significantly higher than for the standard parameters (see Appendix Table 6). However, for SIMON$[12, 5, 3]$ the differential shows a surprisingly different behaviour and the probability of the differential is much closer to the probability of the characteristic. On the other side, the characteristics seem to be worse for the larger variants as can be seen in Table 7. Furthermore it might be desirable to have at least one rotation parameter that corresponds to a byte length, something that the standard parameter set features.

## 7 Conclusion and Future Work

In this work we analysed the general class of functions underlying the SIMON block cipher. First we rigorously derived efficiently computable and easily implementable expressions for the exact differential and linear behaviour of SIMON-like round functions.

Building upon this, we used those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for SIMON. Furthermore, we were able to find all characteristics contributing to the probability of a differential for Simon32 and gave better estimates for the probability for other variants.

Finally, we investigated the space of Simon variants using different rotation constants with respect to diffusion, and the optimal differential and linear characteristics. Interestingly, the default parameters seem to be not always optimal.

This work opens up for further investigations. In particular, the choice and justifications of the NSA parameters for Simon remains unclear. Besides our first progress concerning the round function, the design of the key schedule remains largely unclear and further investigation is needed here.

**Acknowledgements** First of all, we wish to thank Tomer Ashur. Both the method to check whether a linear input mask gives a correlated or uncorrelated linear 1-round characteristic for a given output mask as well as the first version of the SMT/SAT model for linear characteristics in Simon were an outcome of our discussions. We furthermore wish to thank the reviewers for comments that helped to improve the paper.

## References

1. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption, FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 525–545. Springer (2015)
2. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block ciphers - focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014. Lecture Notes in Computer Science, vol. 8616, pp. 57–76. Springer (2014)
3. Alizadeh, J., AlKhzaimi, H., Aref, M.R., Bagheri, N., Gauravaram, P., Kumar, A., Lauridsen, M.M., Sanadhya, S.K.: Cryptanalysis of SIMON variants with connections. In: Saxena, N., Sadeghi, A. (eds.) Radio Frequency Identification: Security and Privacy Issues, RFIDSec 2014. Lecture Notes in Computer Science, vol. 8651, pp. 90–107. Springer (2014)
4. Aumasson, J., Jovanovic, P., Neves, S.: Analysis of NORX: investigating differential and rotational properties. In: Aranha, D.F., Menezes, A. (eds.) Progress in Cryptology - LATINCRYPT 2014. Lecture Notes in Computer Science, vol. 8895, pp. 306–324. Springer (2015)
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), `http://eprint.iacr.org/`
6. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption, FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 546–570. Springer (2015)
7. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012)

8. Carlet, C.: Vectorial boolean functions for cryptography. In: Boolean Models and Methods in Mathematics, Computer Science, and Engineering, Encyclopedia of Mathematics And Its Applications, vol. 134, pp. 398–469. Cambridge Univ. Press (2010)
9. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: The NOEKEON block cipher. Submission to the NESSIE project (2000)
10. Dinur, I., Dunkelman, O., Gutman, M., Shamir, A.: Improved top-down techniques in differential cryptanalysis. Cryptology ePrint Archive, Report 2015/268 (2015), `http://eprint.iacr.org/`
11. Ganesh, V., Hansen, T., Soos, M., Liew, D., Govostes, R.: STP constraint solver (2014), `https://github.com/stp/stp`
12. Grosso, V., Leurent, G., Standaert, F., Varici, K.: LS-designs: Bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption, FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 18–37. Springer (2015)
13. Kölbl, S.: CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives (2015), `https://github.com/kste/cryptosmt`
14. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328 (2013), `http://eprint.iacr.org/`
15. Soos, M.: CryptoMiniSat SAT solver (2014), `https://github.com/msoos/cryptominisat/`
16. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747 (2014), `http://eprint.iacr.org/`
17. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Constructing mixed-integer programming models whose feasible region is exactly the set of all valid differential characteristics of SIMON. Cryptology ePrint Archive, Report 2015/122 (2015), `http://eprint.iacr.org/`
18. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014. Lecture Notes in Computer Science, vol. 8873, pp. 158–178. Springer (2014)
19. Wang, Q., Liu, Z., Varici, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: Meier, W., Mukhopadhyay, D. (eds.) Progress in Cryptology - INDOCRYPT 2014. Lecture Notes in Computer Science, vol. 8885, pp. 143–160. Springer (2014)

## A Short tutorial for calculating differential probabilities and square correlations in SIMON-like round functions

The idea of this section is to complement the rigorous proofs with a more intuitive approach to calculating the differential probabilities and square correlation of one round of Simon. This should also allow us to better understand the Python code given later for calculating these values. We restrict ourselves to a simplified version of the SIMON round function:

$$\begin{aligned} f &: \mathbb{F}_2^n \to \mathbb{F}_2^n && (14) \\ f(m) &= S^1(m) \odot m \ \ . && (15) \end{aligned}$$

Writing this equation bitwise where $m_i$ denotes the $i$th bit of $m$ we obtain:

$$f_i(m) = m_{i-1} \odot m_i \ \ . \tag{16}$$

When using a bit subscript, we will always implicitly assume that the subscript is calculated modulo $n$, the number of bits. Thus $m_{-1}$ and $m_{n-1}$ will for example refer to the same bit.

## A.1 Differential probabilities

Suppose now we are given a message $m = (m_{n-1}, \dots, m_1, m_0)$ and an input difference $d = (d_{n-1}, \dots, d_1, d_0)$. Then the resulting difference $D$ for the function $f$ is calculated as $D(m, d) = f(m) \oplus f(m \oplus d)$. This can be written bitwise as:

$$D_i(m, d) = (m_{i-1} \odot m_i) \oplus ((m_{i-1} \oplus d_{i-1}) \odot (m_i \oplus d_i)) \ \ . \tag{17}$$

By differentiating between the four possible different cases for $d_i$ and $d_{i-1}$, we obtain the following:

$$D_i(m, d) = \begin{cases} 0, & \text{if } d_i = 0 \text{ and } d_{i-1} = 0 \\ m_i, & \text{if } d_i = 0 \text{ and } d_{i-1} = 1 \\ m_{i-1}, & \text{if } d_i = 1 \text{ and } d_{i-1} = 0 \\ \overline{m_i \oplus m_{i-1}}, & \text{if } d_i = 1 \text{ and } d_{i-1} = 1 \ \ . \end{cases} \tag{18}$$

In the last case, $D_i$ is 1 exactly when $m_i = m_{i-1}$ and is 0 when $m_i \neq m_{i-1}$.

Let us now look at a first example. Let $n = 6$, and $d = 001010$. We then calculate $D(m, d)$ using the above bitwise definition of $D$:

$$\begin{array}{l|cccccc} i & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline d & 0 & 0 & 1 & 0 & 1 & 0 \\ S^1(d) & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline D(m, d) & 0 & m_4 & m_2 & m_2 & m_0 & 0 \end{array} \ \ . \tag{19}$$

We can see that the resulting difference depends only on $m_4$, $m_2$ and $m_0$. Thus by adapting these bits appropriately we can generate the following resulting differences:

$$000000, 000010, 001100, 001110, 010000, 010010, 011100, 011110.$$

All these differences then have the same probability of $8/64 = 1/8$. Note that the reuse of a message bit, $m_2$ in this case, is due to a subsequence 101 in the difference.

Let us take a look at another example. Let again $n = 6$ and now $d = 011010$. Then we can again calculate $D(m, d)$ as

$$\begin{array}{lcccccc}
i & 5 & 4 & 3 & 2 & 1 & 0 \\
\hline
d & 0 & 1 & 1 & 1 & 1 & 0 \\
S^1(d) & 1 & 1 & 1 & 1 & 0 & 0 \\
\hline
D(m,d) & m_5 & \overline{m_4 \oplus m_3} & \overline{m_3 \oplus m_2} & \overline{m_2 \oplus m_1} & m_0 & 0
\end{array} . \tag{20}$$

We can see here that consecutive 1s in the input difference will cause the respective output difference bit to depend on two message bits. Nevertheless are all five non-zero output difference bits independent of each other. Thus $2^5$ different output differences are possible, each one with probability 1/32.

With the observations made above, we can now devise a rule that allows us to determine the probability of a given pair $(\alpha, \beta)$ of an input difference $\alpha$ and an output difference $\beta$. First we calculate the set of `varibits` which is the bits in which the output difference can be non-zero. So output bit $\beta_i$ is in `varibits` if and only if $\alpha_i$ or $\alpha_{i-1}$ is non-zero:

$$\texttt{varibits} = \alpha \mid S^1(\alpha) \tag{21}$$

where $\mid$ denotes the bitwise or.

Next we have to calculate which of these output difference bits have to be the same because of patterns 101 in the input difference. We do this by calculating the set `doublebits` which is the output difference bits that always have to be the same as the difference bit one position to the right. Thus $\beta_i$ is in `doublebits` if and only if $\alpha_i$ is 1, $\alpha_{i-1}$ is 0, and $\alpha_{i-2}$ is 1.

$$\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha) \tag{22}$$

To check whether input difference $\alpha$ can map to output difference $\beta$ with non-zero probability, we only need to check whether all non-zero bits of $\beta$ lie in `varibits` and that all bits of $\beta$ that are in `doublebits` are the same as the bits to their right. The probability of the transition is then determine by the number of output difference bits that can be chosen freely, i.e. the number of bits in `varibits` minus the number of bits in `doublebits`.

Before we write this procedure down, we have to take a look at one special case, namely when all input difference bits are set, e.g. $n = 6$ and $d = 111111$. Then we can again calculate $D(m, d)$ as

$$\begin{array}{lcccccc}
i & 5 & 4 & 3 & 2 & 1 & 0 \\
\hline
d & 1 & 1 & 1 & 1 & 1 & 1 \\
S^1(d) & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
D(m,d) & \overline{m_5 \oplus m_4} & \overline{m_4 \oplus m_3} & \overline{m_3 \oplus m_2} & \overline{m_2 \oplus m_1} & \overline{m_1 \oplus m_0} & \overline{m_0 \oplus m_4}
\end{array} . \tag{23}$$

Although all bits of the output are influenced and all bits of the input take equal influence, there are not 64 possible output differences since by switching all bits of $m$ the output difference does not change.

So which output differences are possible then? By fixing an output difference, we get a sequence of equations of the kind $m_i = m_{i+1}$ or $m_i \neq m_{i+1}$. This creates a closed chain of equations that have to be coherent to be satisfiable. As a 0 in the output difference creates an inequality and a 1 creates an equality, in the end it boils down to the condition that the number of 0s in the output difference has to be even when the input difference only consists of 1s.

Let us now summarise all of this in a method to calculate the probability that a given input difference $\alpha$ is mapped to a given output difference $\beta$:

1. Check if $\alpha$ is the difference with all bits set to 1. If that is the case, calculate the number of 0s in $\beta$. If this number is even, return probability $2^{-n+1}$, otherwise return probability 0. If $\alpha$ is not all 1s, go to next step.
2. Calculate `varibits` as $\texttt{varibits} = \alpha \mid S^1(\alpha)$. Check if $\beta$ has any bits set to 1 which are not in `varibits`, i.e. check if $\overline{\texttt{varibits}} \odot \beta \neq \mathbf{0}$. Should this be the case, return probability 0. Otherwise continue with next step.
3. Calculate `doublebits` as $\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha)$. Check whether there are any bits of $\beta$ in `doublebits` that are not equal to their right neighbour, i.e. check $\left(\beta + S^1(\beta)\right) \odot \texttt{doublebits} \neq \mathbf{0}$. Should this be the case, return probability 0. Otherwise continue with next step.
4. Return probability $2^{-\operatorname{wt}(\texttt{varibits}+\texttt{doublebits})}$.

This method allows us to determine differential probabilities of the function $f(x) = S^1(x) \odot x$. We only have to apply some affine transformation to convert this to a method for calculating the probability of the Simon round function. A Python implementation of the more general method can be found in Section B.

### A.2 Square correlations

Let us now look at how to calculate square correlations for $f(x) = S^1(x) \odot x$.

First we look at the case where the input mask $\alpha$ is all 0s. Let $n = 6$ and let the output mask $\beta$ be 010110:

$$\begin{array}{l|cccccc} \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline m & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\ S^1(m) & m_4 & m_3 & m_2 & m_1 & m_0 & m_5 \\ \hline \beta & 0 & 1 & 0 & 1 & 1 & 0 \end{array} . \tag{24}$$

The resulting expression is then

$$m_4 m_3 + m_2 m_1 + m_1 m_0. \tag{25}$$

Let us look at the first term. It is zero in 3 out of 4 cases. It thus has a correlation of $\frac{1}{2}$ and hence a square correlation of $\frac{1}{4}$.

Let us look at the next two terms $m_2 m_1$ and $m_1 m_0$. First we note that they are not independent as they share the variable $m_1$. So we cannot calculate the square correlation of the sum of these terms as the product of the square correlations of the single terms. But we can rewrite the sum of these terms as

$$m_2m_1 + m_1m_0 = m_1(m_2 + m_0). \tag{26}$$

Now $(m_2 + m_0)$ behaves like a single one bit variable. Therefore the square correlation of $m_1(m_2+m_0)$ is $\frac{1}{4}$ as well. As $m_4m_3$ and $m_1(m_2+m_0)$ do not share any variables, the square correlation of the whole expression $m_4m_3+m_2m_1+m_1m_0$ is then $\frac{1}{4} \cdot \frac{1}{4} = \frac{1}{16}$. It is easy to see that different "blocks" of 1s in $\beta$ that are separated by at least one 0, will generate independent terms. We thus only need to look at the square correlations of the terms generated from these blocks and multiply these to get the final result.

Let us thus look at a longer block of 1s with $\beta = 011111$:

$$\begin{array}{lcccccc}
\alpha & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
m & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\
S^1(m) & m_4 & m_3 & m_2 & m_1 & m_0 & m_5 \\
\hline
\beta & 1 & 1 & 1 & 1 & 1 & 0
\end{array}\ . \tag{27}$$

The resulting expression is

$$m_5m_4 + m_4m_3 + m_3m_2 + m_2m_1 + m_1m_0. \tag{28}$$

By combining the first and the second as well as the third and the fourth term, we get

$$m_4(m_5 + m_3) + m_2(m_3 + m_1) + m_0m_1. \tag{29}$$

As $(m_5 + m_3)$, $(m_3 + m_1)$, and $m_1$ are independent of each other, the three terms $m_4(m_5 + m_3)$, $m_2(m_3 + m_1)$, and $m_0m_1$ are independent and the square correlation of the whole expression is thus $\left(\frac{1}{4}\right)^3 = 2^{-6}$.

At this point we already dare to formulate a rule. The square correlation of the term generated by $m$ consecutive blocks of 1s is $2^{-2\lceil \frac{m}{2} \rceil}$. As every pair of consecutive single terms can be combined to create one independent term of square correlation $2^{-2}$, the total square correlation just depends on the number of terms left after such pairing. And this number is $\lceil \frac{m}{2} \rceil$.

Let us now consider a non-zero input mask $\alpha$. Let $\alpha = 010010$ and let $\beta = 010100$:

$$\begin{array}{lcccccc}
\alpha & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
m & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\
S^1(m) & m_4 & m_3 & m_2 & m_1 & m_0 & m_5 \\
\hline
\beta & 0 & 1 & 0 & 1 & 0 & 0
\end{array}\ . \tag{30}$$

The resulting expression is then

$$m_4m_3 + m_4 + m_2m_1 + m_2 + m_1 + m_0. \tag{31}$$

We see that we can combine the first two terms to get a term of square correlation $2^{-2}$ again: $m_4m_3 + m_4 = m_4(m_3 + 1)$. Note that if the second term had been $m_3$

instead, it would have worked too. For the next three terms we can do the same: $m_2m_1 + m_2 + m_1 = (m_2 + 1)(m_1 + 1) + 1$. Note that the bias of this term is now flipped; the square correlation is nonetheless also $2^{-2}$. As the first two terms are independent of the next three terms, the square correlation of the combined first five terms is $2^{-4}$. But when looking at the last term $m_0$, we see that it is independent of all other terms and unbiased. Thus the square correlation of the complete expression is 0.

As a matter of fact, it is easy to see that when for any $i$ the respective bit $\alpha_i$ of the input mask is 1 but both $\beta_i$ and $\beta_{i+1}$ are 0, the resulting expression will always be unbiased. Thus we can say that every non-zero bit in the input mask belonging to some block of 1s in the output mask is a necessary condition for the whole expression to be unbiased. Note that every bit in the input mask can at most be associated with one block of 1s in the output mask.

Thus we can evaluate the square correlation of $f$ for an input mask $\alpha$ and an output mask $\beta$ like this: First we check whether every non-zero bit in the input mask is associated to a block of 1s in the output mask. Is this not the case, we already know that the square correlation is zero. Otherwise we continue to partition the output mask and the input mask into blocks of 1s and their associated input mask bits. For each of these blocks we determine the square correlation of the resulting expression and finally multiply these together to get the total square correlation.

But how do we evaluate a block of 1s with the associated input mask bits in general? In the last example, we saw that for a block of a single 1 in the output mask, the two associated bits of the input mask can take any value; the square correlation remains $2^{-2}$.

How about in the case of a block of two 1s? Let us look at the case of $\alpha = 111001$ and let $\beta = 110110$:

$$\begin{array}{lcccccc}
\alpha & 1 & 1 & 1 & 0 & 0 & 1 \\
\hline
m & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\
S^1(m) & m_4 & m_3 & m_2 & m_1 & m_0 & m_5 \\
\hline
\beta & 1 & 1 & 0 & 1 & 1 & 0
\end{array}\ . \tag{32}$$

The resulting expression is

$$m_5m_4 + m_4m_3 + m_5 + m_4 + m_3 + m_2m_1 + m_1m_0 + m_0. \tag{33}$$

Let us first look at the first block of 1s in the output mask $\beta$, i.e. at the expression $m_5m_4 + m_4m_3 + m_5 + m_4 + m_3$. Combining the first two terms, we get

$$m_4(m_5 + m_3) + m_5 + m_4 + m_3. \tag{34}$$

We can now combine the first term with $m_4$ to get

$$m_4(m_5 + m_3 + 1) + m_5 + m_3. \tag{35}$$

Finally we can also incorporate $m_5$ and $m_3$ to get

$$(m_4 + 1)(m_5 + m_3 + 1) + 1. \tag{36}$$

The expression thus has a square correlation of $2^{-2}$.

Let us look at the expression generated by the second block of 1s in the output mask:

$$m_2m_1 + m_1m_0 + m_0. \tag{37}$$

Combining the first two terms, we get

$$m_1(m_2 + m_0) + m_0. \tag{38}$$

But now we see that the term $m_0$ is independent of the first term. Thus we are left with a square correlation of 0. Note that the square correlation would also be 0 if the last term were $m_2$ but not if the last term were $m_1$ in which case the square correlation would be $2^{-2}$.

As a matter of fact, the rule to determine the square correlation of an expression generated by a block of two 1s in the output mask and the associated bits in the input mask is straightforward. There are three associated input mask bits. If and only if both or none of the two outer bits ($m_2$ and $m_0$ in the last example) are set to 1, is the expression biased and the square correlation is $2^{-2}$.

In fact, for a block of an even number of 1s in the output mask, any combination of associated input bits, will lead to a biased expression with the same square correlation. For a block of an odd number of 1s in the output mask, we need to check the input mask though. There is an odd number of associated bits to this block in the input mask. Let us refer to the first bit and then every second bit as the odd bits, and to the second bit and then every second bit as the even bits (from which direction we count does not matter). The even bits do not have an influence on the square correlation. But the parity of the odd bits determines whether the expression for this block will be unbiased or not. If and only if the parity is even, the expression is biased.

We can summarise a method to calculate the square correlation for a given input mask $\alpha$ and a given output mask $\beta$ that is not all 1s as follows:

1. Partition the 1s in the output mask into consecutive blocks of 1s. The total square correlation is now the product of the square correlations for each block.
2. For each block calculate the square correlation:
   (a) If the block length is odd, this block is always biased and the square correlation is solely determined by its length.
   (b) If the block length is even, we need to check the input mask. There is an odd number of bits in the input mask that are associated with this output block. Calculate the XOR of every second bit of these associated bits starting with the first one (such that both outer bits are considered). If this XOR sum is 1, the block is unbiased and thus the whole expression is unbiased. If the XOR sum is 0, the square correlation for this block is determined by its length.

For an implementation of the method to calculate the square correlation in Python, see Section B.

## B Python code to calculate differential probabilities and square correlations in SIMON-like round functions

In the following, code for calculating the differential probabilities and square correlations of SIMON-like round functions ($f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x)$) are given in Python. Restrictions are that the constants need to fulfil $\gcd(a-b, n) = 1$. We assume that the functions $S^d(x)$ and $\mathrm{wt}(x)$ have been implemented as well as a function `parity` that calculates the parity $\mathrm{wt}(x) \mod 2$ of a bit vector $x$. $a$, $b$, and $c$ have to be defined in the program as well.

The differential probability of $\alpha \xrightarrow{f} \beta$ can then be calculated with the following function:

```python
def pdiff (alpha,beta):
    # Use gamma instead of beta to get rid of linear part
    gamma = beta ^ S(alpha,c)
    # Take care of the case where alpha is all 1s
    if alpha == 2**n-1:
        if hw(~gamma)%2 == 0:
            return 2**(n-1)
        else:
            return 0
    # Determine bits that can take a nonzero difference
    varibits = S(alpha, a) | S(alpha,b)
    # Check whether gamma conforms with varibits
    if gamma & ~varibits != 0:
        return 0
    # Determine the bits that are duplicates
    doublebits = S(alpha,2*a-b) & ~S(alpha,a) & S(alpha,b)
    # Check whether the duplicate bits are the same as there counterpart
    if (gamma ^ S(gamma,a-b)) & doublebits != 0:
        return 0
    return 2**(-hw(varibits^doublebits))
```

The squared correlation of $\alpha \xrightarrow{f} \beta$ can be calculated with the following function. Here we assume $n$ to be even, which is relevant for the case where $\beta$ is all 1s.

```python
def plin (alpha,beta):
    # Get rid of linear part of round function
    alpha ^= S(beta,-c)
    # If the input masks uses bits that have corresponding bits
    # in the output mask, the correlation is 0.
    if ((S(beta,-a) | S(beta,-b)) ^ alpha) & alpha != 0:
        return 0
    # Take care of the case where beta is all 1s
    if beta == 2**n-1:
           t, v = alpha, 0
           while t != 0:
               v ^= t & 3
               t >>= 2
           if v != 0:
               return 0
           else:
               return 2**(-n+2)
    # Set in the abits mask the first and then every second bit of each
    # block of 1s in the output mask beta. Each corresponds to one
    # independent multiplication term, and thus adds a factor of 2^(-2)
    # to the square correlation.
    # Example: beta = 0111101110110 -> abits = 0101001010100
    tmp = beta
    abits = beta
```

```
while tmp != 0:
    tmp = beta & S(tmp, -(a-b))
    abits ^= tmp
# The sbits correspond to bits one to the right of each block of an
# even number of 1s in the output mask.
# Example: beta = 0111101110110 -> sbits = 0000010000001
sbits = S(beta, -(a-b)) & ~beta & ~S(abits, -(a-b))
# Adopt sbits to correspond to the respective bits in the input
# mask
sbits = S(sbits, -b)
# The pbits are used to check whether the input mask removes the
# bias from one of the output mask blocks. It checks the parity of
# the sum of every second inputmask bit for each block that
# corresponds to a block of an even number of 1s in the output mask.
pbits = 0
while sbits != 0:
    pbits ^= sbits & alpha
    sbits = S(sbits, (a-b)) & S(beta,-b)
    sbits = S(sbits, (a-b))
    pbits = S(pbits, 2*(a-b))
# If the parity is uneven for any one of the blocks, there is no bias.
if pbits != 0:
    return 0
return 2**(-2*hw(abits))
```

## C Additional Differential Bounds

In Table 4 resp. 5 we give the distributions for the characteristics contributing to a differential up to the bound we computed them.

## D Optimal parameters for differential characteristics

The following sets of rotation constants $(a, b, c)$ are optimal for 10 rounds regarding differential characteristics for Simon32, Simon48, and Simon64

$$(1,0,2),(1,0,3),(2,1,3),(4,3,5),(5,0,10),(5,0,15),(5,4,3),(7,0,14),(7,6,5)$$
$$(8,1,3),(8,3,14),(8,7,5),(10,5,15),(11,6,1),(12,1,7),(12,5,3),(12,7,1)$$
$$(13,0,10),(13,0,7),(13,8,2)$$

Similar to the experiments for the default parameters, we used our framework to evaluate the quality of various rotation constants. In Table 7 we give an overview of the best differential characteristics for variants of Simon using a different set of rotation constants. Table 6 shows that a carefully chosen set of constants can have a very strong effect on the differentials.

**Table 4.** Number of differential characteristics for the differential $(80, 222) \xrightarrow{f^{17}} (222, 80)$ for Simon48.

| $\log_2(p)$ | #Characteristics | $\log_2(p)$ | #Characteristics |
|---|---|---|---|
| −52 | 1 | −69 | 20890 |
| −53 | 6 | −70 | 38837 |
| −54 | 15 | −71 | 72822 |
| −55 | 46 | −72 | 133410 |
| −56 | 100 | −73 | 240790 |
| −57 | 208 | −74 | 353176 |
| −58 | 379 | −75 | 279833 |
| −59 | 685 | −76 | 235071 |
| −60 | 1067 | −77 | 259029 |
| −61 | 1607 | −78 | 225836 |
| −62 | 2255 | −79 | 256135 |
| −63 | 2839 | −80 | 252193 |
| −64 | 3476 | −81 | 252654 |
| −65 | 4088 | −82 | 198784 |
| −66 | 5032 | −83 | 229843 |
| −67 | 7063 | −84 | 208757 |
| −68 | 11481 | −85 | 253112 |

**Table 5.** Number of differential characteristics for the differential $(4000000, 11000000) \xrightarrow{f^{21}} (11000000, 4000000)$ for Simon64.

| $\log_2(p)$ | #Characteristics | $\log_2(p)$ | #Characteristics |
|---|---|---|---|
| −68 | 2 | −83 | 185709 |
| −69 | 14 | −84 | 173860 |
| −70 | 70 | −85 | 171902 |
| −71 | 276 | −86 | 171302 |
| −72 | 951 | −87 | 168190 |
| −73 | 2880 | −88 | 164694 |
| −74 | 8101 | −89 | 163141 |
| −75 | 21062 | −90 | 161089 |
| −76 | 52255 | −91 | 159354 |
| −77 | 123206 | −92 | 155804 |
| −78 | 238297 | −93 | 150954 |
| −79 | 239305 | −94 | 145061 |
| −80 | 171895 | −95 | 141914 |
| −81 | 170187 | −96 | 138480 |
| −82 | 165671 | −97 | 132931 |

**Table 6.** Distribution of the characteristics for a 13-round differential for SIMON32 using different set of constants.

| $\log_2(p)$ | $[8,1,2]$ | $[12,5,3]$ | $[7,0,2]$ | $[1,0,2]$ |
|---|---|---|---|---|
| −36 | 1 | 1 | 4 | 1 |
| −37 | 4 | 2 | 16 | 6 |
| −38 | 15 | 3 | 56 | 27 |
| −39 | 46 | 2 | 144 | 88 |
| −40 | 124 | 1 | 336 | 283 |
| −41 | 288 | 0 | 744 | 822 |
| −42 | 673 | 0 | 1644 | 2297 |
| −43 | 1426 | 0 | 3420 | 6006 |
| −44 | 2973 | 0 | 6933 | 14954 |
| −45 | 5962 | 0 | 13270 | 34524 |
| −46 | 11661 | 1 | 24436 | 73972 |
| −47 | 21916 | 3 | 43784 | 150272 |
| −48 | 40226 | 14 | 76261 | 292118 |
| −49 | 72246 | 32 | 130068 | / |
| −50 | 126574 | 54 | 218832 | / |
| −51 | 218516 | 83 | 362284 | / |

**Table 7.** Overview of the optimal differential characteristics for SIMON variants.

| Rounds: | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Differential** $(12,5,3)$ | | | | | | | | | | | | | | | |
| SIMON32 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −28 | −34 | −36 | −42 | −44 | −47 |
| SIMON48 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −36 | −38 | −40 | −42 |
| SIMON64 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −35 | −37 | −43 | −47 | / |
| **Differential** $(1,0,2)$ | | | | | | | | | | | | | | | |
| SIMON32 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −36 | −38 | −40 | −42 |
| SIMON48 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −48 | −54 |
| SIMON64 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −48 | −54 |
| **Differential** $(7,0,2)$ | | | | | | | | | | | | | | | |
| SIMON32 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −25 | −30 | −35 | −36 | −38 | −40 | −42 |
| SIMON48 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −35 | −38 | −44 | −48 | −53 |
| SIMON64 | −2 | −4 | −6 | −8 | −12 | −14 | −18 | −20 | −26 | −30 | −36 | −38 | −44 | −48 | / |

**Table 8.** For each SIMON variant and each possible number of rounds, the number of possible combinations of rotation constants $(a, b, c)$ with $a \geq b$ is given that reaches full diffusion.

SIMON32

| Rounds | 6 | 7 | 8 | 9 | 10 | 11 | 17 | $\infty$ |
|---|---|---|---|---|---|---|---|---|
| $\#(a, b, c)$ | 48 | 600 | 528 | 88 | 144 | 128 | 64 | 576 |

SIMON48

| Rounds | 7 | 8 | 9 | 10 | 11 | 13 | 14 | 15 | 25 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\#(a, b, c)$ | 48 | 1392 | 1680 | 792 | 528 | 344 | 144 | 128 | 64 | 2080 |

SIMON64

| Rounds | 8 | 9 | 10 | 11 | 12 | 13 | 15 | 17 | 18 | 19 | 33 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\#(a, b, c)$ | 384 | 4800 | 2112 | 2256 | 1152 | 608 | 512 | 48 | 288 | 256 | 128 | 4352 |

SIMON96

| Rounds | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 19 | 21 | 25 | 26 | 27 | 49 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\#(a, b, c)$ | 336 | 4272 | 13920 | 7104 | 5568 | 3456 | 912 | 1152 | 800 | 1568 | 640 | 48 | 288 | 256 | 128 | 16000 |

SIMON128

| Rounds | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 23 | 24 | 25 | 33 | 34 | 35 | 65 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\#(a, b, c)$ | 768 | 10944 | 26112 | 25536 | 9024 | 6912 | 7488 | 2496 | 192 | 1824 | 2304 | 1792 | 1024 | 960 | 512 | 96 | 576 | 512 | 256 | 33792 |