J. Cryptol. (2014) 27: 452–479
DOI: 10.1007/s00145-013-9150-0

Journal of
CRYPTOLOGY

Rotational Rebound Attacks on Reduced Skein

Dmitry Khovratovich

University of Luxembourg, Luxembourg, Luxembourg
dmitry.khovratovich@uni.lu

Ivica Nikolić

Nanyang Technological University, Singapore, Singapore
inikolic@ntu.edu.sg

Christian Rechberger

DTU, Lyngby, Denmark
christian.rechberger@groestl.info

Communicated by Meier

Received 27 March 2012
Online publication 31 May 2013

Abstract. In this paper we combine two powerful methods of symmetric cryptanalysis: rotational cryptanalysis and the rebound attack. Rotational cryptanalysis was designed for the analysis of bit-oriented designs like ARX (Addition-Rotation-XOR) schemes. It has been applied to several hash functions and block ciphers, including the new standard SHA-3 (Keccak). The rebound attack is a start-from-the-middle approach for finding differential paths and conforming pairs in byte-oriented designs like Substitution-Permutation networks and AES.

We apply our new compositional attack to the reduced version of the hash function Skein, a finalist of the SHA-3 competition. Our attack penetrates more than two thirds of the Skein core—the cipher Threefish, and made the designers to change the submission in order to prevent it.

The rebound part of our attack has been significantly enhanced to deliver results on the largest number of rounds. We also use neutral bits and message modification methods from the practice of collision search in MD5 and SHA-1 hash functions. These methods push the rotational property through more rounds than previous analysis suggested, and eventually establish a distinguishing property for the reduced Threefish cipher. We formally prove that such a property cannot be found for an ideal cipher within the complexity limits of our attack. The complexity estimates are supported by extensive experiments.

Key words. Skein, SHA-3, Hash function, Compression function, Cipher, Rotational cryptanalysis, Rebound attack, Distinguisher.

1. Introduction

Block ciphers and hash functions are cornerstones of symmetric cryptography, where privacy and authenticity of communication are established by efficient schemes. Design and analysis of these primitives was pushed by the competitions for new standards: most notably AES for block ciphers (1998–2001) and SHA-3 for hash functions (2008–2012). Not only new designs, but also new types of attacks emerged from these two competitions.

The SHA-3 competition emerged from the concerns among the cryptographic community of the new attacks on hash functions that had appeared by 2005 [6,39,44,45]. The standard SHA-2 followed the same design strategies as broken SHA-0 and SHA-1, and it was largely unknown if the new attacks could be carried out to SHA-2 within the next years. The American standardization organization NIST after a series of workshops decided to run a public competition for a new standard, hoping to come up with a replacement earlier than SHA-2 would be under attack as well. Whereas attacks on SHA-2 progressed slowly since 2006 [16,24,31,33,37], NIST eventually chose Keccak [4] out of 64 submissions.

Among the methods proposed for analysis of the SHA-3 candidates, rotational cryptanalysis and the rebound attack are notably universal and effective. Rotational analysis is well suited for bit-oriented designs, in particular for those based on modular addition, rotation, and XOR (so-called ARX schemes). The reduced versions of SHA-3 candidates Skein [22], Shabal [1], BMW [38], and eventually the SHA-3 winner Keccak [11] are affected by this type of attack, despite their relative resistance to the well-studied differential cryptanalysis.

The rebound attack, first presented in [34], was initially and mostly aimed at byte-oriented primitives with an SPN structure. It produces conforming pairs for differential paths in a meet-in-the-middle fashion, essentially shaving off the most "expensive" part of a differential path. The method gives the best results so far on reduced variants of the SHA-3 candidates Grøstl [17] and ECHO [18], LANE [30], Luffa [21], Cheetah [46] and the hash function Whirlpool [26], among others. It also yields a differential distinguisher for the largest number of rounds of SHA-3 (Keccak) [11].

Our Results In this paper we combine the rotational and the rebound attacks with the application to the compression function of the SHA-3 candidate Skein and its underlying cipher Threefish in the version 1.2 [14]. We carefully use the degrees of freedom in the inbound phase of the rebound attack, so that we attack many more rounds compared to all other results on Skein/Threefish. We introduce a new type of distinguishing property, called a rotational collision, and prove formally that in the black-box model the complexity of finding such collisions is significantly higher than the complexity of producing rotational collisions for the Skein-256 compression function reduced up to 53 (out of 72) rounds, and for the Skein-512 compression function reduced up to 55 (out of 72) rounds. Our approach is aimed for the largest number of rounds at the cost of complexity, but similar results with almost practical complexity can be drawn for the reduced number of rounds (out of this paper's scope). We also provide a more accurate estimation of rotational probabilities compared to [22].

Our results demonstrate weaknesses both in the reduced Threefish cipher (in the ideal cipher model) and in the Skein compression function as of version 1.2. Our models require that the key, the message, and the tweak can be freely chosen by an attacker, which is certainly not the case for a hash function or block cipher. Nevertheless, our attacks show that reduced Threefish should not instantiate an ideal cipher in any indifferentiability proof using it, like the one used for the Skein hash function [13]. The designers eventually responded to our attack by changing Skein so that its components are much less vulnerable to the rotational analysis [15].

Details on Our Rotational-Rebound Attack We start our research with a more detailed and careful analysis of the rotational property and its propagation. We represent it analytically, and derive necessary conditions on the key bits to increase the rotational probability and thus reduce the complexity of our attacks. We also correct [22] in terms of the independence assumptions, and find the best values of the key bits with an optimized computer search. Although we attack the second version of Skein (v1.2 [14]), we would like to stress out that our attack approach is applicable to the first version of Skein as well, but does not apply to the latest version of Skein-v1.3.

This preliminary rotational analysis gives us a rotational distinguisher for the compression function of Skein on up to 40 rounds. We advance further and show how to put the rotational property into the outbound phase of the *rebound attack*. The inner part of the rebound attack, which is the inbound phase, is accelerated with the method of the auxiliary path [19] and neutral bits [5]. In contrast to the first attacks on Skein, where these paths were used in differential attacks, we demonstrate their use in the rotational attack. As a result, we get a rotational distinguisher for the reduced Skein compression function. We attack 53 rounds of Skein-256 and 55 rounds of Skein-512 (Sect. 5), whereas the full versions have 72 rounds.

2. Preliminaries

2.1. *Short History of Skein*

Skein is a family of hash functions designed by Ferguson et al. [12]. Since its submission to the SHA-3 competition in 2008, Skein underwent a series of revisions. The first revision, yielding the version 1.1, appeared quickly after the submission and corrected typos. The second revision (version 1.2) appeared in September 2009 and allegedly improved the diffusion properties with a new set of rotation constants [14]. Our analysis, as well as the first conceptual paper on the rotational cryptanalysis [22], was published in the first half of 2010 and is devoted to this version.

As soon as the next phase of the SHA-3 competition again allowed changes, the designers of Skein responded to the rotational attacks and tweaked it to the version 1.3 in October 2010 [15]. The tweak solely changed the constant in the key schedule, which efficiently prohibits rotational cryptanalysis. Though Skein was considered a favorite in the competition till its end, NIST eventually declared Keccak as the winner on October 2, 2012.

Table 1. Summary of the attacks on Skein and Threefish.

Rounds	Version	Attack	Method	Reference
Skein/Threefish-256-256 (72 rounds)				
24	1.1	Key recovery	Related-key differential	Design document, 2008
24	1.2	Near collision	Differential characteristic	Su et al. [43], 2010
39	1.2	Key recovery	Related-key rotational	Khovratovich–Nikolić, [22], 2010
53	1.2	Rotational collision	Chosen-key-tweak Rotational rebound	This paper
12	1.3	Collisions (hash)	Biclique	Khovratovich [20], 2012
24	1.3	Near-collision	Differential characteristic	Leurent [27], 2012
32	1.3	Diff. distinguisher	Known-related-key boomerang	Leurent–Roy, [28], 2012
32	1.3	Partial collision	Differential characteristic	Leurent [27], 2012, Yu et al. [47], 2013
Skein/Threefish-512-512 (72 rounds)				
25	1.1	Key recovery	Related-key differential	Design document, 2008
32	1.1	Key recovery	Related-key boomerang	Aumasson et al. [2], 2009
35	1.1	Diff. distinguisher	Known-related-key boomerang	Aumasson et al. [2], 2009
24	1.2	Near collision	Differential characteristic	Su et al. [43], 2010
34	1.2	Key recovery	Related-key boomerang	Chen-Jia [9], 2010
42	1.2	Key recovery	Related-key rotational	Khovratovich–Nikolić, [22], 2010
55	1.2	Rotational collision	Chosen-key-tweak Rotational rebound	This paper
14	1.3	Collisions (hash)	Biclique	Khovratovich [20], 2012
22	1.3	Preimage (hash)	Biclique	Savelieva et al. [24], 2012
37	1.3	Free-start collision	Biclique	Li et al. [29], 2012
37	1.3	Preimage (compression)	Biclique	Savelieva et al. [24], 2012

2.1.1. *Third-Party Cryptanalysis of Skein*

The first third-party cryptanalysis [2] targeted various properties of the Skein v1.1 compression function and Threefish up to 35 rounds. Near-collisions up to 24 rounds were investigated in [43]. The first rotational attack demonstrated distinguishers on the underlying Threefish cipher [22], and was able to penetrate 39 rounds of Skein-256 v1.2 and 42 rounds of Skein-512 v1.2 (see Table 1).

After a preliminary version of this paper has been published [23], two more results have been announced. Biclique preimage attacks [24] cryptanalyze 22 rounds of the Skein-512 v1.3 hash function and 37 rounds of the Skein-512 compression function. Advanced boomerang distinguishers were applied on up to 32 rounds of Threefish v1.3 [28], and new differential near-collision attacks on the Skein v 1.3 compression function for up to 32 steps were proposed in [27,47].

2.2. *Description of Skein v1.2*

Skein is a family of hash functions, which are based on the different versions of the block cipher Threefish. It has three incarnations: with 256-, 512-, and 1024-bit block and the same key sizes. The 1024-bit version was not intended for the SHA-3 output

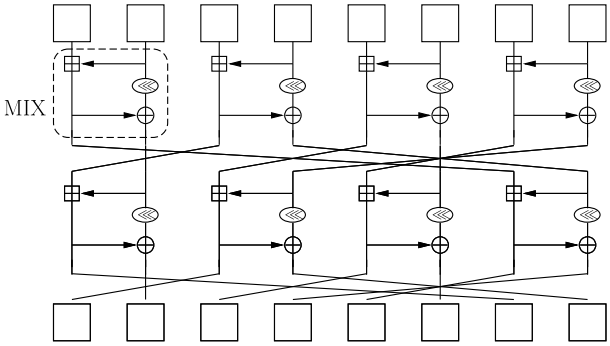

Fig. 1. Two rounds of Threefish-512.

sizes (224, 256, 384, 512 bits), and later it was explicitly stated that only Threefish-512 would be used for all output sizes of the purposes of SHA-3 competition. In this work, we analyze Threefish-256 and Threefish-512.

Threefish is a tweakable block cipher, where the tweak value T is a public 128-bit input, and could be used to further parametrize the cipher and to break the similarity between the compression function calls when used in Skein. By $E_{K,T}(P)$ we denote Threefish with input key K, a tweak T, and a plaintext P. The compression function F of Skein uses Threefish in the Matyas–Meyer–Oseas (MMO) mode:

$$F_T(CV, M) = E_{CV,T}(M) \oplus M, \tag{1}$$

where CV is the chaining value, and M is the message block of the same size. The hash function of Skein produces the output after a sequence of compression function calls. As our analysis does not go beyond the compression function, and as the full description of the hash function is rather complicated, we omit it and refer to [15].

The definition of the cipher is as follow. The internal state I undergoes a sequence of 72 similar rounds, and after each fourth round a subkey is modularly added to the state. An additional subkey addition (key whitening) is done at the beginning of the first round.

Internal Round Each round has a simple structure (cf. also Fig. 1). The internal state is partitioned into N_w ($N_w = 4, 8$ for Threefish-256,-512, respectively) 64-bit words $I_0, I_1, \ldots, I_{N_w-1}$. Then two distinct operations are applied to all the state words. The first is a pairwise non-linear MIX operation, while the second is a simple word permutation π:

Round r, $0 \le r < 72$:

1. For $0 \le j < N_w/2$ set
   - $(I_{2j}, I_{2j+1}) \leftarrow \mathrm{MIX}((I_{2j}, I_{2j+1}))$;
2. For $0 \le j < N_w$ set
   - $I_j^{\mathrm{new}} \leftarrow I_{\pi(j)}$.

The operation π depends on the round number r, while MIX depends as well on the index j, and the output $(Y_1, Y_2) = \mathrm{MIX}(X_1, X_2)$ is defined as follows:

$$Y_1 = X_1 + X_2,$$
$$Y_2 = (X_2 \lll R_{(r \bmod 8),j}) \oplus Y_1.$$

The rotation constants $R_{r,j}$ and the permutation π are defined in Appendix A.

Key Schedule and Subkeys Let $K_0, K_1, \ldots, K_{N_w-1}$ be the 64-bit words of the master key K. An additional word K_{N_w}, acting as a checksum, is computed as

$$K_{N_w} = \texttt{0x55}\ldots\texttt{5} \oplus \bigoplus_{j=0}^{N_w-1} K_j.$$

Here we would like to point out that the word rotation (either to the left or to the right) by an even amount of bits does not change the constant `0x55...5`—this is a crucial property exploited in our subsequent rotational attack. In the final version of Threefish, i.e. v1.3, the constant was changed to `0x1BD11BDAA9FC1A22`.

Similarly, a checksum tweak word T_2 is computed from the 64-bit tweaks words T_0, T_1, i.e. $T_2 = T_0 \oplus T_1$. Let $K^0, K^1, \ldots, K^{18}$ be the subkeys, and $K^s_0, K^s_1, \ldots, K^s_{N_w-1}$ be 64-bit words of the subkey K^s, $s = 0, \ldots, 18$. These words are computed as follows:

$$K^s_j = K_{(s+j) \bmod (N_w+1)}, \quad 0 \le j \le N_w - 4;$$
$$K^s_{N_w-3} = K_{(s+N_w-2) \bmod (N_w+1)} + T_{s \bmod 3};$$
$$K^s_{N_w-2} = K_{(s+N_w-1) \bmod (N_w+1)} + T_{(s+1) \bmod 3};$$
$$K^s_{N_w-1} = K_{(s+N_w) \bmod (N_w+1)} + s.$$

Note the counter s added to the last subkey word.

3. Rotational and Rebound Attacks

3.1. *Rotational Cryptanalysis*

It has been known for a while that if one vector is a rotation of the other, then the bitwise operations such as XOR or AND keep this property. Some designers used this fact in the initial analysis of their own cryptosystems [3,41], whereas Daum explored the propagation of this property through the modular addition operation [10]. The term *rotational cryptanalysis* was introduced by Khovratovich and Nikolić in the analysis of Skein [22].

The pair $(X, \overleftarrow{X})$ is called a *rotational pair* (with a rotation amount r), where $\overleftarrow{X}$ the rotation of X by r bits to the left. A rotational pair is preserved by any bitwise transformation, particularly by the bitwise XOR and by any rotation. The probability that a rotational pair is kept by the modular addition is given by the following formula [10]:

$$\mathbf{P}\left(\overleftarrow{x+y} = \overleftarrow{x} + \overleftarrow{y}\right) = \frac{1}{4}\left(1 + 2^{r-n} + 2^{-r} + 2^{-n}\right). \tag{2}$$

For large n and small r we get the following table:

r	p_r	$\log_2(p_r)$
1	0.375	−1.415
2	0.313	−1.676
3	0.281	−1.831

For $r = n/2$ the probability is the lowest and it is close to 1/4. The same holds for rotations to the right. When an addition of rotational inputs does not produce rotational outputs then we say that the addition produces a *rotational error*.

Similarly to differential and linear cryptanalysis, the rotational cryptanalysis requires the rotational property to hold through a number of rounds of a primitive. Clearly, the probability of this event depends on the number of operations that may violate the rotational property, e.g., the modular addition.

The simplest rotational attack first establishes that for the primitive F with n-bit output the *rotational property holds* with probability $\mathbb{P} \gg 2^{-n}$:

$$F\left(\overleftarrow{X}\right) \overset{\mathbb{P}}{=} \overleftarrow{F(X)},$$

for some pre-fixed rotational amount r.

It will be proven in Sect. 4 that this property and its variations yield non-random behavior for unkeyed primitives like compression and hash functions. In the further text, we always work with a single rotational amount, an optimal value of which has to be found. For keyed primitives this property allows for shortcut key recovery attacks, as it can be used as a distinguisher to verify partial key guesses.

Constant Addition The use of constants is typical countermeasure against slide attacks and other methods exploiting similarity of rounds. The addition of a constant also violates the rotational property unless the constant is self-rotational, i.e. if $C = \overleftarrow{C}$ then $\overleftarrow{X} \oplus C = \overleftarrow{X \oplus C}$. However, if the constant addition follows another operation that may fail to preserve the rotational pair, then the resulting errors may compensate each other. The first rotational attack on Skein [22] made the errors introduced by three consecutive operations to compensate each other (see Fig. 2).

3.2. *Rebound Attack*

The rebound attack [26,34] was introduced as a variant of differential cryptanalysis optimized for the cryptanalysis of hash functions. It aims to efficiently produce inputs conforming to valid differential paths. At the same time, the rebound attack can be seen as a high-level model for the cryptanalysis of key-less primitives. It was first applied to AES-like constructions because it is easy to find truncated differential characteristics in them for a number of rounds. Distinguishers for generic Feistel schemes [40] and meet-in-the-middle attacks on block ciphers [8] also use elements of the rebound attack.

The rebound attack (Fig. 3) decomposes primitive E—a compression function, a block cipher, or a permutation—into three parts:

$$E = E_3 \circ E_2 \circ E_1.$$

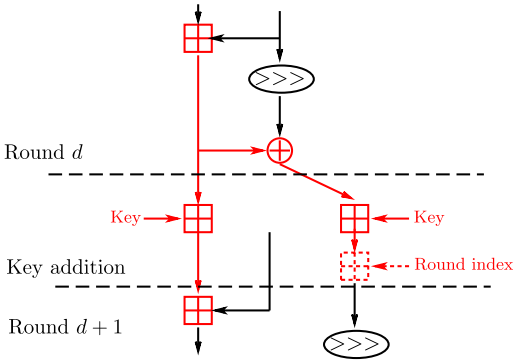

Fig. 2. Dealing with constant addition in Threefish: error is introduced by the modular addition, then is corrected by the key addition, and is finally compensated by another modular addition.

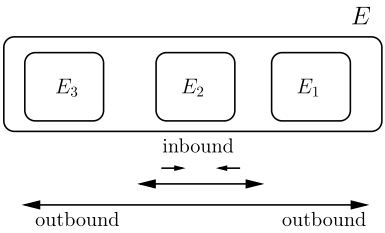

Fig. 3. Outline of the rebound attack.

The two phases are as follows:

- **Inbound phase** searches for inputs conforming to some property (usually, to a differential path) in the meet-in-the-middle fashion in E_2. Here the search is efficiently aided by the degrees of freedom available to a cryptanalyst.
- **Outbound phase** computes the solutions of the inbound phase in both forward- and backward direction through E_1 and E_3 and checks whether they are solutions for the full E. If this is a probabilistic event, an attacker repeats the inbound phase to obtain more starting points for the outbound phase.

Recent modifications of the rebound approach include the inside-out variant [32], the linear solving variant [32], or the multiple-inbound variant [26,30].

Our idea is to target the rotational property in the rebound attack, so that the inputs conforming to the rotational property in the inbound phase can be found with a low complexity. In the next section we formally prove that our approach produces outputs which are improbable to find for an ideal primitive with the same complexity.

4. Rotational Distinguishers

In this section we argue that the attack, described in detail in Sect. 5, indeed shows non-random behavior of the Skein compression function. A typical argument would

show that an attacker with only a black-box access to an ideal primitive of the same domain and range is not able to produce the same behavior with the same or better effort and probability. We follow the approach of [7], where the adversary produces so-called q-multicollisions for AES significantly faster than for an ideal cipher. Then we carry over this statement to the compression function.

The Threefish key schedule uses a counter in each subkey K^s. As none of these counters are rotation-invariant, the subkey injection always violates the rotational property of a pair of internal states. As indicated in Sect. 3.1, we have to compensate emerging errors by other probabilistic operations around the subkey injection. As will be explained in details in Sect. 5, adding a constant e to the chaining value CV is sufficient to obtain the rotational property with reasonable probability for a reduced Skein compression function F:

$$F_{\overleftarrow{T}}\left(\overleftarrow{CV}+e,\overleftarrow{M}\right)\overset{\mathbb{P}}{=}\overleftarrow{F_T(CV,M)}. \tag{3}$$

Moreover, we can produce q such inputs with average complexity $1/\mathbb{P}$ and the same e:

$$\begin{cases}\overleftarrow{F_{T_1}(CV_1,M_1)}=F_{\overleftarrow{T_1}}\left(\overleftarrow{CV_1}+e,\overleftarrow{M_1}\right);\\ \overleftarrow{F_{T_2}(CV_2,M_2)}=F_{\overleftarrow{T_2}}\left(\overleftarrow{CV_2}+e,\overleftarrow{M_2}\right);\\ \vdots\\ \overleftarrow{F_{T_q}(CV_q,M_q)}=F_{\overleftarrow{T_q}}\left(\overleftarrow{CV_q}+e,\overleftarrow{M_q}\right).\end{cases} \tag{4}$$

Let us introduce an appropriate definition of the wanted property.

Definition 1. A set

$$\left\{e;(CV_1,M_1,T_1),(CV_2,M_2,T_2),\ldots,(CV_q,M_q,T_q)\right\}$$

is called *a rotational q-collision set* for a compression function $F_T(CV,M)$ if (4) holds for it.

A similar definition can be introduced for the cipher on which the compression function is based. The MMO mode (1) yields the following conversion:

$$\overleftarrow{F_T(CV,M)}=F_{\overleftarrow{T}}\left(\overleftarrow{CV}+e,\overleftarrow{M}\right)\quad\Longleftrightarrow\quad\overleftarrow{E_{CV,T}(M)}=E_{\overleftarrow{CV}+e,\overleftarrow{T}}\left(\overleftarrow{M}\right).$$

Hence we can introduce an appropriate definition for a tweakable cipher.

Definition 2. A set

$$\left\{e;(P_1,K_1,T_1),(P_2,K_2,T_2),\ldots,(P_q,K_q,T_q)\right\}$$

is called *a rotational q-collision set* for a tweakable cipher $E_{K,T}(P)$ if

$$\begin{cases} \overleftarrow{E_{K_1,T_1}(P_1)} = E_{\overleftarrow{K_1}+e,\overleftarrow{T_1}}\left(\overleftarrow{P_1}\right); \\ \overleftarrow{E_{K_2,T_2}(P_2)} = E_{\overleftarrow{K_2}+e,\overleftarrow{T_2}}\left(\overleftarrow{P_2}\right); \\ \vdots \\ \overleftarrow{E_{K_q,T_q}(P_q)} = E_{\overleftarrow{K_q}+e,\overleftarrow{T_q}}\left(\overleftarrow{P_q}\right). \end{cases} \tag{5}$$

We follow the line of the first distinguisher for the full AES [7] and compare the problem of finding a rotational collision set for an ideal cipher with that for reduced Threefish. By ideal cipher, as usual, we understand a set of randomly chosen permutations of cardinality equal to the size of the key space. Our results demonstrate that the versions of Threefish that we consider do not behave like an ideal cipher with respect to this rotational property. Afterwards we proceed with the same statement on the compression function.

The complexity of the generic attack is measured in the number of queries to the encryption and decryption oracles of an ideal cipher.

Lemma 1. *To construct a rotational q-collision set for an ideal (tweakable) cipher with an n-bit block and key and success rate $1/2$, an adversary needs at least* $\min(\frac{q}{12} \cdot 2^{((q-1)/(q+1))n}, 2^{n-1})$ *queries.*

Proof. Let A be an adversary attacking the cipher, and assume that A asks its oracles a total of L queries, where $L < 2^{n-1}$. Let us compute the probability of the event that a rotational q-collision set (5) is found. The probability is taken over all possible choices of permutations for the cipher.

First, we denote the equations in (5) as $U_1, U_2, \ldots, U_q$. With each equation we associate an integer t_j such that t_jth oracle query computes the chronologically second element of U_j and hence is able to check whether the equation holds. Without loss of generality, assume that $t_1 < t_2 < \cdots < t_q$. Finally, define t'_1 as the index of the query that computes the first element of the equation U_1:

$$\begin{cases} U_1: \underbrace{\overleftarrow{E_{K_1,T_1}(P_1)}}_{\text{queried at } t'_1} = \underbrace{E_{\overleftarrow{K_1}+e,\overleftarrow{T_1}}\left(\overleftarrow{P_1}\right)}_{\text{queried at } t_1}; \\ U_2: \underbrace{\overleftarrow{E_{K_2,T_2}(P_2)}}_{\text{queried before } t_2} = \underbrace{E_{\overleftarrow{K_2}+e,\overleftarrow{T_2}}\left(\overleftarrow{P_2}\right)}_{\text{queried at } t_2}; \\ \vdots \\ U_q: \underbrace{\overleftarrow{E_{K_q,T_q}(P_q)}}_{\text{queried before } t_q} = \underbrace{E_{\overleftarrow{K_q}+e,\overleftarrow{T_q}}\left(\overleftarrow{P_q}\right)}_{\text{queried at } t_q}. \end{cases} \tag{6}$$

Now compute for every tuple $(t'_1, t_1, t_2, t_3, \ldots, t_q)$ the probability that it leads to a differential q-multicollision. Before submitting t_ith query, $i > 1$, equations

$U_1, U_2, \ldots, U_{i-1}$ hold, where terms of $U_1, U_2, \ldots, U_{i-1}$ are completely determined by a tuple $(t'_1, t_1, t_2, t_3, \ldots, t_{i-1})$. Indeed, from t'_1 and t_1 we define K_1, e, P_1, T_1, the rotation amount; from t_j we define K_j, T_j, and P_j.

Just before the moment t_i only one term of U_i is computed—w.l.o.g. let it be $E_{K_i,T_i}(P_i)$. Thus the following equation should hold:

$$\overleftarrow{E_{K_i,T_i}(P_i)} = \underbrace{E_{\overleftarrow{K_i}+e,\overleftarrow{T_i}}\left(\overleftarrow{P_i}\right)}_{\text{queried at } t_i}.$$

By our definition, t_i is the first moment when $E_{\overleftarrow{K_i}+e,\overleftarrow{T_i}}(\overleftarrow{P_i})$ is queried. Then either the decryption or the encryption oracle is called. In the first case the decryption oracle is called with a ciphertext C and a key K, which for some i should be equal to $\overleftarrow{K_i} + e$. By the definition of t_i, the value C is chosen from the set where $E_{\overleftarrow{K_i}+e,,\overleftarrow{T_i}}(\cdot)$ is undefined. To become a part of a rotational q-collision set, there should exist P_i such that $C = \overleftarrow{E_{K_i,T_i}(P_i)}$. On the other hand, after the decryption oracle is called, the following equation should hold:

$$E^{-1}_{\overleftarrow{K_i}+e,\overleftarrow{T_i}}(C) = \overleftarrow{P_i}. \tag{7}$$

Since $L < 2^{n-1}$, not more than 2^{n-1} texts were encrypted or decrypted with the key $\overleftarrow{K_i} + e$. So the probability that (7) holds does not exceed $1/2^{n-1}$.

In the second case, let the encryption oracle be queried with a plaintext P, tweak T, and a key K, which for some i should be equal to $\overleftarrow{K_i} + e$. For an answer C, a similar equation should hold:

$$C = \overleftarrow{E_{K_i,T_i}(P_i)}. \tag{8}$$

The same probability argument holds for this equation. Therefore, for every t_i, $i \geq 2$, we get a multiplier 2^{1-n} to the probability that a tuple $(t'_1, t_1, t_2, t_3, \ldots, t_q)$ defines a rotational q-collision set. There are $\binom{L}{q+1}$ such tuples, each defining a rotational q-collision set with probability at max $2^{(q-1)(1-n)}$. We get the following equation for the number of queries required to get a q-collision set with probability $1/2$:

$$\binom{L}{q+1} \geq 2^{(q-1)(n-1)-1}. \tag{9}$$

Let us simplify the left part:

$$\begin{aligned}\binom{L}{q+1} &= \frac{L!}{(L-q-1)!(q+1)!} = \frac{L(L-1)\cdots(L-q)}{(q+1)!} \\ &\leq \frac{L^{q+1}}{(q+1)!} \leq \frac{L^{q+1}}{\frac{(q+1)^{q+1}}{e^{q+1}}} = \left(\frac{eL}{q+1}\right)^{q+1}.\end{aligned} \tag{10}$$

Substitute the result to (9):

$$\left(\frac{eL}{q+1}\right)^{q+1} \geq 2^{(q-1)(n-1)-1} \quad \Longrightarrow \quad L \geq \frac{q+1}{e} 2^{((q-1)/(q+1))(n-1)-1} \geq \frac{q}{12} 2^{((q-1)/(q+1))n}. \tag{11}$$

This concludes the proof. □

Let us consider compression functions now. An ideal compression function is introduced similarly to the ideal cipher and is an equivalent of a random PRF. So, an ideal compression function over a particular domain with a given range is a set of randomly chosen transformations with the same domain and range.

Theorem 1. *To construct a rotational q-collision set for an ideal compression function with an n-bit output and success rate $1/2$, an adversary needs at least* $\min(\frac{q}{12} \cdot 2^{((q-1)/(q+1))n}, 2^{n-1})$ *queries.*

Proof. The proof is almost identical to the proof of lemma 1. However, we are equipped with a single-compression oracle, and do not perform any sort of decryption. Hence we merely omit from the proof the "first case" where the decryption oracle is called. Given the bound of 2^{n-1} queries, we find that for every tuple of query indices of size $q+1$ the probability that it defines a rotational q-collision set does not exceed $2^{(q-1)(1-n)}$. The rest of the proof remains the same. □

In the next section we show how to obtain a rotational q-collision set for reduced Threefish and the Skein compression function.

5. Rotational Rebound Attack on the Skein Compression Function

5.1. *Overview*

Our goal in this section is to construct a rotational q-collision set for the cipher Threefish, which immediately converts to a rotational q-collision set for the Skein compression function. We proceed as follows:

- Fix the optimal rotation amount;
- Find and fix the optimal key values K_i;
- Calculate the transition probabilities and demonstrate that there exist inputs for which the rotational property holds throughout reduced Threefish (details in Sect. 5.3);
- Identify the rounds for the inbound phase, where states conforming to the rotational property can be generated efficiently, and show the procedure;
- Identify neutral bits that help to ensure the rotational property beyond the inbound phase;
- Identify remaining degrees of freedom and estimate the total complexity of the attack including the outbound phase;
- Demonstrate that for some q the attack outputs rotational q-collisions faster than what the lower bound for the ideal case instructs.

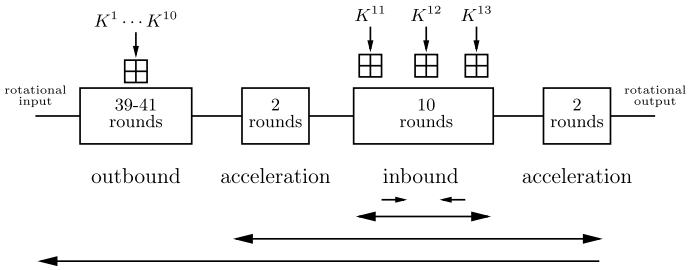

Fig. 4. The complete rotational rebound attack on Threefish-256, -512. *Arrows* indicate the direction of the computation.

Table 2. Structure of the rebound attack on Skein.

Outbound		Acceleration I	Inbound	Acceleration II
Rounds	Probability	Rounds	Rounds	Rounds
Skein-256 (53 rounds)				
2–40	2^{-239}	41–42	43–52	53–54
Skein-512 (55 rounds)				
0–40	2^{-480}	41–42	43–52	53–54

Having all these details elaborated, the *attack* would proceed as follows:

1. Produce internal states that conform to the rotational property through the inbound phase;
2. Filter out those for which the rotational property does not hold in the rounds of the acceleration phase;
3. Generate more solutions for the rounds covered in the acceleration phase;
4. Filter out those that do not conform to the rotational property through the outbound phase.

An illustration of the attack proposal is given Fig. 4, while also given in Table 2.

Eventually for the fixed correction e and rotational amount r we produce tuples (P, K, T) such that

$$E_{\overleftarrow{K}+e,\overleftarrow{T}}(\overleftarrow{P}) = \overleftarrow{E_{K,T}(P)},$$

where E is the Threefish-256 reduced to rounds 2–54 (0–50 for the 512-bit version), without the encompassing key addition. For the Skein compression function, this yields tuples (CV, T, M) such that

$$F_{\overleftarrow{T}}\left(\overleftarrow{IV} + e, \overleftarrow{M}\right) = \overleftarrow{F_T(IV, M)}$$

for the same e. The total complexity is about 2^{239} per tuple in Skein-256, and 2^{480} per tuple in Skein-512. Here and further the complexity unit is one evaluation of the compression function. The memory consumption is about 2^{30} Skein states. Having fixed $q = 2^6$ for both variants, we are able to construct a rotational q-collision set for the

Skein compression function with complexity lower than for an ideal compression function. Also, we can construct a rotational q-collision set for the cipher Threefish with complexity lower than for an ideal cipher. This proves the distinguishing nature of our attack.

5.2. *Selecting Parameters for the Attack*

Rotation Amount We recalled in Sect. 3.1 that the rotation by 1 bit delivers the highest probability for the modular addition. However, the constant `0x55...5` is invariant to the rotation by 2 bits. Since the other constants used in Threefish—round counters—are not rotation-invariant for any amount, we select the rotation amount as two bits to the left.

Corrections Our experiments showed that the counter in the key schedule quite often prohibits the rotational property to hold for several consecutive rounds. If there were no corrections, the probability for the rotational property to hold through rounds $4s-1, 4s$ (i.e. with the key addition in between) would be zero for quite many s. To avoid that, we fix some of the key bits and introduce corrections. As a result, the errors introduced by counters, corrections, and modular additions compensate each other, and we want this property to hold as long as possible.

Optimal key bit and the resulting correction values were the subject of our experiments and were found with an optimized computer search. The optimal values are given in Tables 3 and 4. Hence the inbound phase of the rebound attack starts with assigning the 24 bits (for Threefish-256) or 48 bits (for Threefish-512) of the key K and its counterpart $\overleftarrow{K}+e$ with actual values.

5.3. *Rotational Probabilities*

This section gives an outline of the search for optimal key values. More details are given in Appendix B.

We follow the idea of [22], and introduce corrections in the Threefish key pair. However, unlike [22], we consider modular corrections, i.e. we define the related-key pair as $(K, \overleftarrow{K}+e)$, where e is a low-weight correction and $+$ is a modular addition. The rotation amount is fixed to 2 in order make the constant used in the key schedule, and thus the checksum master key word K_{N_W}, susceptible to rotational analysis, and also to maximize the rotational probability of modular additions.[1]

To obtain the highest number of rounds in the outbound phase, we find optimal values for the corrections and well as values of several bits of the key pair. These values are found with an exhaustive search on a computer. However, due to the large size of the search space, a simple brute force would be infeasible, and thus first we have to significantly reduce the amount of possible candidates by performing a detailed analysis. Further we explain how to optimize the search in Skein-256 (see Fig. B.1, the rotational pairs are presented one atop of another).

[1] A rotation amount of 4 (or any other larger even amount) would also make the checksum words susceptible to rotational analysis, but would reduce the rotational probability of modular additions, see (2).

Table 3. Pre-fixed values of key bits for the rotational pair and the decimal value of the correction in Skein-256. The middle 58 bits of K_i coincide (regarding rotation) in K and its rotated counterpart.

	K_0	K_1	K_2	K_3	K_4
K	0111..10	0100..11	0011..10	0000..11	0101..01
$\overleftarrow{K} + e$	11..0011	00..1010	11..0110	00..1001	01..0011

First we divide the cipher into pairs of consecutive rounds. There are two types of such pairs. The first type is composed of pairs that do not have a subkey addition in between the rounds, e.g. rounds 5 and 6, rounds 9 and 10, etc. As such double rounds have no operations involving counters (there is no subkey addition), we assume all the input and output pairs of these rounds to be fully rotational. Thus their rotational probability is fixed to $2^{-8.5}$ for Skein-256 and 2^{-17} for Skein-512. These number were obtained empirically with computer experiments and in fact differ from the theoretical values of $2^{-6.7}$ (4 modular additions in the two rounds, each with rotational probability $2^{-1.67}$) and $2^{-13.4}$ used in [22]. The second type of double rounds consists of pairs of consecutive rounds of Skein-256 that have a subkey addition in between (such as rounds 3 and 4, 7 and 8, etc.). Only such pairs could be used to efficiently prune and optimize the search, and to find the optimal values for the corrections and the key bits. The details of our search are quite technical, and we describe them in Appendix B.

We assigned optimal values to $6 \times 4 = 24$ bits of the first master key, i.e. 4 MSBs and 2 LSBs of each 64-bit word, and 24 bits of the second key. The values of these bits are given in Table 3. Once we had the optimal values of the keys and the optimal differences, we found the probability for four consecutive rounds. We start with a random rotational input pair of states and go through three rounds. Then we add the subkeys (with the particular counters) and then we go for an additional round. The outcome of this testing is given at Table C.2 of Appendix C. Thus, in Skein-256 the probability to pass rounds 2–41 (i.e. 10 key additions) is about 2^{-239}.

Skein-512 Optimal values for the differences and some key bits can be obtained for Skein-512 as well. A property of the double subkey rounds of Skein-512 that helps to run the optimal search is that these two double subkey rounds can be split into two non-overlapping halves (see Fig. C.1 in Appendix C), and then for each half the optimal differences can be found independently. Note that this simply speeds up the search for optimal differences and values, and has no impact on the actual probability of the rotational property. Unlike Skein-256, in Skein-512 we could not find empirically the probabilities for four consecutive rounds because they were too low. Instead, we considered each four rounds as double round + double subkey round and simply multiplied the probabilities of these two. The values for the optimal 6 bits of each key word in Skein-512 are given in Table 4. In Skein-512 the probability to pass rounds 0–41 is about 2^{-480} (details in Table C.3).

5.3.1. *Probabilities in the Khovratovich–Nikolić Analysis*

The paper [22] provided the rotational analysis of Threefish on up to 42 rounds. The probability estimates were based on several independence assumptions, which must be corrected as follows:

Table 4. Pre-fixed values of key bits and correction in Skein-512.

	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K	0111..10	0100..01	0011..10	0000..10	0111..01	0000..01	0011..10	0000..10	0001..10
$\overleftarrow{K} + e$	11..0011	00..0010	11..0010	00..0001	11..0011	00..0010	11..0010	00..0001	01..0101

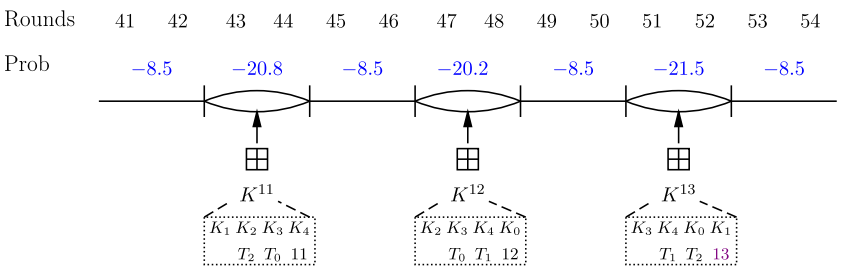

Fig. 5. Probabilities in the inbound phase for Skein-256.

- The probability of the rotational pair propagation through double rounds without key addition (2–3, 6–7, etc.) is not a multiplication of probabilities for a single round. The problem is that two consecutive modular additions $((a \boxplus b) \boxplus c)$ have lower rotational probability than expected. For example, the rotational probability of one round in Skein-256 is $2^{-3.35}$ for the rotation by 2, but the probability of two rounds is $2^{-8.52}$ instead of $2^{2\cdot(-3.35)} = 2^{-6.7}$.
- The rotational inputs to the round before the key addition (4, 8, etc.) are not uniformly distributed, and this partly compensates the negative effect of the dependency (see above). We note that the non-uniformity of inputs is best approximated with restricting the two most significant bits to the value {00}.
- The propagation of the rotational inputs through the double round with the key addition in Threefish-256, with the appearance and the correction of errors, can not be considered as two independent events (i.e., as getting rotational pairs in the further MIX operations independently). As a result, the probability of this event can not be computed as a multiplication of other probabilities, and must be computed as a single value.

5.4. *Inbound Phase*

We are going to produce a pair of states, keys, and tweaks, that follow the rotational trail in rounds 43–52. The rotational probability for Skein-256 is equal to $2^{-79.5}$. We show how to produce a conforming input with negligible amortized cost. Please refer to Fig. 5 for more details. These rounds are chosen as the most expensive for the rotational property, which makes the forward direction of the outbound phase very short. Nevertheless, our attack could be equally well run with the inbound phase in the middle at the cost of a slightly increased complexity.

First, we produce 2^{30} states that conform to the rotational trail in rounds 45–46, and do the same for rounds 49–50. This can be done with negligible amortized cost, as we basically need to fix a handful of bits to ensure the rotational property to propagate

through modular additions. Then we match those states by determining the value of subkey K^{12}. We have already estimated that with probability $2^{-20.2}$ the resulting subkey compensates the errors introduced by the counter. Hence we output $2^{30+30-20}$ solutions for rounds 45–50 with complexity 2^{60}.

The subkey K^{12} determines the words $K_2, K_3 + T_0, K_4 + T_1, K_0$. We can freely choose K_1 and K_3 in order to pass through rounds 51–52 and 43–44. We note that only the least significant bits affect the rotational property when computing rounds 51–52 in the forward direction. Having fixed the least 20 significant bits for K_1 and K_3, we can filter out the states not conforming to rounds 51–52. Hence we are left with 2^{19} solutions for rounds 45–52 and 80 bits of freedom left (note the pre-fixed key bit values). Now let us note that the knowledge of $K_2, K_3 + T_0, K_4 + T_1, K_0$ and 20 LSBs of K_1 and K_3 determines 20 LSBs of subkey K^{11}. This allows to compute back the rounds 43–44 except for the modular additions in round 43. This allows to filter out almost all internal states incompatible with the conditions of rounds 43–44, with about 5 filtering bits left. Hence we produce $2^{19-20+5} = 2^4$ states that conform to all but five bit conditions in rounds 43–52, and still have 80 bits of freedom left. Hence the amortized cost of building a solution for the full section of rounds 43–52 is about 2^5. The solutions for Skein-512 are built in a similar way. The memory consumption is about 2^{30} Skein states.

In the next section we shall see how to further exploit the degrees of freedom we apparently have left.

5.5. *Acceleration Phase*

The acceleration phase of the attack may be seen as part of the inbound phase or part of the outbound phase. Technically, starting from here computations are done in an inside-out manner, yet remaining degrees of freedom are used to accelerate the search for right pairs in the outbound phase.

As soon as we get a right pair of computations for the inbound phase, we produce many more of them from the given one as follows. We follow the simple idea of neutral bits as e.g. applied in the analysis of SHA-0 and SHA-1 [5]. We view them as auxiliary path [19] (also formalized as tunnels or submarines in [25,36,42]) and apply the differences specified by the path to the key and the tweak.

The configuration of the auxiliary path for Skein-256 is given in Table 5. We apply the original path difference to the first execution of the pair, and the rotated path difference to the second execution.

We consider $\oplus$-differences here, so we have to take into account the fact that the tweak and the key are added by the modular addition. Therefore, we choose the difference so that the probability of observing a carry is low. However, since adjacent bits are often neutral as well, a carry bit may still preserve the rotational pair.

In Skein-256 we take various δ and apply the resulting auxiliary path $\mathcal{P}_\delta$ to the right pair. We choose δ so that the differences in the subkey K^{12} compensate each other. Then we check whether the modular additions in rounds 41–42 and 53–54 are not affected by the modification. If so, we get another rotational pair for rounds 41–54.

In experiments, we found that 44 of the 64 possible individual bits that result in a local collision of the latter type behave neutral with probability larger than 0.75 for three rounds in forward direction and simultaneously two rounds in backwards direc-

Table 5. Configuration of the auxiliary path for Skein-256. K_i is the ith word of the first subkey K^0.

Round	Subkey	Master key words			
43–44	K^{11}	K_1 0	K_2 0	K_3 δ	K_4 δ
		Tweak words			
			T_2 0	T_0 δ	
47–48	K^{12}	K_2 0	K_3 δ	K_4 δ	K_0 0
			T_0 δ	T_1 δ	
51–52	K^{13}	K_3 δ	K_4 δ	K_0 0	K_1 0
			T_1 δ	T_2 0	

tion, 37 consecutive bits of those have a probability very close[2] to 1. Details for this phase will be found in Appendix in Table C.1. Overall, the results mean that every time those four rounds in the outbound phase are computed, and the effort of those is less than 2^{37}, the amortized effort for those computations will be negligible. If the effort for those five rounds is more, the effect of this acceleration phase, the speed-up, still grows roughly exponentially with the number of neutral bits used.

5.6. *Degrees of Freedom Analysis*

Now we discuss the following question: How often can this inbound phase be repeated? After fixing the differences and the corrections, for Skein-256 we have $256 + 256 + 128 = 640$ degrees of freedom available to perform the attack. The outbound phase fixes 24 of the 256 bits of the key (also 12 bits of the 128-bit tweak), and in addition may need up to 256 bits to follow the longest possible trail with high probability. What remains is $640 - 36 - 256 = 348$ degrees of freedom to be spent by the inbound and the acceleration phase. In Skein-512 we would have $512 + 512 + 128 = 1152$ degrees of freedom, of which $1152 - 512 - 60 = 580$ bits are left. If variants with less rounds are targeted, this number is higher, as less repetitions are needed for the shorter outbound phase. Overall, this is enough for our purposes.

5.7. *Summary and Complexity Estimates*

We experimentally verified the probabilities of the outbound phase, and took various dependencies into account, and also experimentally verified parts of the acceleration and inbound phase.

[2] The fact that carries have to behave equivalently for round key additions in both forward and backward direction puts constraints on the inbound phase which are ignored here to keep the exposition simple. This either results in fewer degrees of freedom available to perform the exhaustive-search part of the attack, or reduces the number of possible combinations of neutral bits, and has to be taken into account in the overall estimate of the time complexity.

Using the Skein-256 compression function as an example, we describe the resulting attack. As illustrated already in Fig. 4, the 8-round inbound part is performed close to the output of the cipher/compression function, the 4 round acceleration area (2 rounds in each direction) surrounding it. The majority of the inside-out computation is then done in backwards direction, covering 38 rounds for Skein-256 and 40 rounds for Skein-512. In total this gives about 52/54 rounds. Additionally, early stopping techniques will only require the computation of a small number of rounds in the outbound part before another trial is made, saving a factor of the computational complexity that is in the order of the number of rounds.

We estimate the amortized cost for the rounds covered by inbound and acceleration phase for both Skein-256 and Skein-512 by a computation that is equivalent to a single computation of the compression function, as there are plenty of neutral bits that cover up costs in solving the right pairs in those inner rounds. In Skein-256, we will spend 2^{239} computations in the outbound+acceleration phases to find 2^{239} starting pairs for the outbound phase. One such pair will pass this phase with probability close to one. Therefore with an effort that is roughly equivalent to 2^{239} calls to the compression function of Skein-256 we can find one rotational pair of messages and chaining values (with corrections) that produces a rotational pair of updated chaining values. To produce 2^6 such pairs, i.e. to find 2^6-rotational collisions in Skein-256, we only need $2^{6+239} = 2^{245}$ calls. On the other hand, in an ideal function one has to make at least $2^{2.5} \cdot 2^{\frac{64-1}{64+1}256} \approx 2^{250}$ calls (see Lemma 1).

Similarly, for the compression function of Skein-512, we can create 2^6-rotational collision set with $2^{6+480} = 2^{486}$ compression function calls, while an ideal function would require $2^{2.5} \cdot 2^{\frac{63}{65}512} \approx 2^{499}$ calls.

5.8. *Probabilities with the New Key Schedule*

Skein v1.3 differs from v1.2 in the constant used to generate the subkey word K_{N_w}. As a result, rotated key will not generate rotated subkeys: every fifth word in the subkey sequence would violate the rotational property. As a result, most of key addition layers would generate additional rotational errors. We expect those errors to vanish with probability not higher than 2^{-32}, which subtracts at least 20 rounds from the Skein-256 attack, and 15 rounds from the Skein-512 attack, ignoring possible troubles in the inbound phase. As a result, we do not pursue our attacks for the new version of Skein.

6. Conclusion and Future Work

Our results do not threaten the practical use of full-round Skein or Threefish. However, we show that reduced versions of these constructions behave in a non-random manner in settings where all or most of the inputs could be chosen, and this holds for many more rounds than initially expected. We argue that variants of Threefish reduced from 72 to about 52/54 rounds, in the chosen-key-and-tweak model, do not behave like an ideal cipher with respect to the rotational property we have defined. Remember that the ideal cipher model implies that the key is freely chosen, and hence nothing is said about the security of Threefish as a PRP. For the compression function of Skein a similar

argument is made. Due to the finalization round, our results are unlikely to carry over to the actual hash function.

To summarize, the following ideas and approaches lead to the improved results:

- The rebound approach as a high-level model for the attack.
- Considering rotational corrections with respect to integer addition instead of XOR.
- Based on analytic reasoning, we find an efficient search method for fixing a subset of input bits before other phases of attacks.
- Using the degrees of freedom in the internal state to efficiently solve for the inner 8-rounds.
- Using the 8-round local collision as long-range neutral bits in an inside-out manner to speed up the outbound phase.

It will be interesting to study how rotational properties found in other constructions, some of which have been reported recently (for SHA-3 e.g. in [35]), can also be amplified in a way similar to what we demonstrated in this paper for Skein. Our new methods cannot directly be used to recover key bits in Threefish in a secret-key model—this is another open problem. The inbound and acceleration techniques we use in our analysis are to a large extent independent of the statistical property that is meant to be produced at the inputs and outputs of Skein. Hence, in addition to the rotational attacks described in this paper, also more traditional differential attacks aiming for collision or near-collision attacks will be able to take advantage of those techniques.

Acknowledgements

This work was sponsored in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the European Commission under contract ICT-2007-216646 (ECRYPT II). Ivica Nikolić is supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

We greatly thank the reviewers of the Journal of Cryptology for their detailed comments, which helped to improve the presentation quality of this paper. The reviewers also helped to identify vague definitions and observations and bring more formalism and clarity to the results. The final version carries new figures, attack and methods overviews, and a revisited description of the rebound part of the attack. The authors would gratefully acknowledge future comments on their research.

Appendix A. Skein Constants

The permutation π is defined as follows:

i	0	1	2	3	4	5	6	7
$\pi(i)$: Threefish-256	0	3	2	1				
$\pi(i)$: Threefish-512	2	1	4	7	6	5	0	3

The rotation constants $R_{r,j}$ are defined as follows:

$r \setminus j$	Threefish-256		Threefish-512			
	0	1	0	1	2	3
0	14	16	46	36	19	37
1	52	57	33	27	14	42
2	23	40	17	49	36	39
3	5	37	44	9	54	56
4	25	33	39	30	34	24
5	46	12	13	50	10	17
6	58	22	25	29	39	43
7	32	32	8	35	56	22

Appendix B. Search for Optimal Corrections

Notations Each 64-bit word w used in Skein can be seen as a concatenation of two words w_1, w_2, i.e. $w = w_1||w_2$ where w_1 represent the two most significant bits of w and w_2 the rest 62 bits. Obviously, this representation is chosen to comply with the rotation amount of 2, i.e. $\overleftarrow{w} = w_2||w_1$. Further we use this representation to give a definition of the words used in the double rounds (refer to Fig. B.1). In fact, *the brute force search could be performed on all possible values of the smaller 2-bit parts, and the optimized search would even reduce the space of possible values, usually from* $2^2 = 4$ *to only 2.*

Further we introduce notations that follow the one used in Fig. B.1. We denote the four words of the internal state at the beginning of the double rounds with (A, B, C, D), and thus the rotational pair of inputs is

$$(A, B, C, D) = \big(a_1||a_2, b_1||b_2, s_1||s_2, t_1||t_2\big);$$
$$\big(\overleftarrow{A}, \overleftarrow{B}, \overleftarrow{C}, \overleftarrow{D}\big) = \big(a_2||a_1, b_2||b_1, s_2||s_1, t_2||t_1\big).$$

The rotational pair of subkeys added after the first round is denoted as

$$K = \big[k_1||k_2, k_3||k_4, k_5||k_6, k_7||k_8\big]; \qquad \overleftarrow{K} + e = \big[k'_2||k'_1, k'_4||k'_3, k'_6||k'_5, k'_8||k'_7\big].$$

Then the corrections $e = [e_0, e_1, e_2, e_3]$ can be computed as

$$e_i = k'_{2i+1}||k'_{2i+2} - k_{2i+1}||k_{2i+2}.$$

Here the tweak value is already added to the subkey K. The rotational states are separated with a horizontal dashed line $(- - - - -)$.

For r-bit words $z_1, \ldots, z_k$ we define the carry $C_{z_1,\ldots,z_k}$ of the sum $z_1 + \cdots + z_k$, i.e. $C_{z_1,\ldots,z_k} = (z_1 + \cdots + z_k) \ggg_r$. We omit specifying the precise value of r (it will be either 2 or 62), and assume it is clear from the context. Finally, we introduce the variables $r, v, \mathcal{D}, \mathcal{U}, x, f$ to maintain the $2 + 62$ bit representation of the words, and with $i = i_1||i_2$ we denote the round counter.

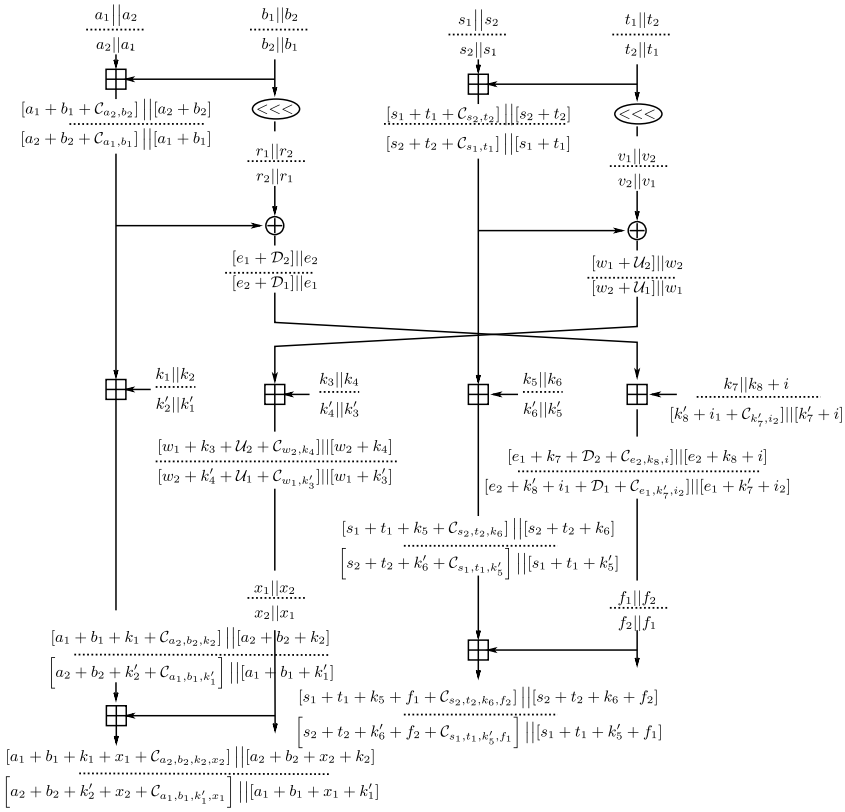

Fig. B.1. Rotational pair through two rounds with key addition of Skein-256.

Reducing the Search Space Since the rotation and XOR preserve the rotational property with probability 1, we avoid investigating these operations in the second round of the double subkey rounds. Thus it is sufficient to obtain rotational pairs just before these operations. To obtain such pairs for the first output (the left most word in Fig. B.1), the following conditions have to hold:

$$\begin{aligned} a_1+b_1+k_1+x_1+C_{a_2,b_2,k_2,x_2} &= a_1+b_1+x_1+k_1', \\ a_2+b_2+x_2+k_2 &= a_2+b_2+k_2'+x_2+C_{a_1,b_1,k_1',x_1} \end{aligned}$$

Similarly, for the rest three outputs, we get the following conditions:

$$\begin{aligned} w_1+k_3+\mathcal{U}_2+C_{w_2,k_4} &= w_1+k_3', \\ w_2+k_4 &= w_2+k_4'+\mathcal{U}_1+C_{w_1,k_3'}, \\ s_1+t_1+k_5+f_1+C_{s_2,t_2,k_6,f_2} &= s_1+t_1+k_5'+f_1, \\ s_2+t_2+k_6+f_2 &= s_2+t_2+k_6'+f_2+C_{s_1,t_1,k_5',f_1}, \\ e_1+k_7+\mathcal{D}_2+C_{e_2,k_8,i} &= e_1+k_7'+i_2, \\ e_2+k_8+i &= e_2+k_8'+i_1+\mathcal{D}_1+C_{e_1,k_7',i_2} \end{aligned}$$

The above eight equations can be reduced to

$$k'_1 - k_1 = C_{a_2,b_2,k_2,x_2}, \tag{B.1}$$

$$k'_2 - k_2 = -C_{a_1,b_1,k'_1,x_1}, \tag{B.2}$$

$$k'_3 - k_3 = C_{w_2,k_4} + \mathcal{U}_2, \tag{B.3}$$

$$k'_4 - k_4 = -(C_{w_1,k'_3} + \mathcal{U}_1), \tag{B.4}$$

$$k'_5 - k_5 = C_{s_2,t_2,k_6,f_2}, \tag{B.5}$$

$$k'_6 - k_6 = -C_{s_1,t_1,k'_5,f_1}, \tag{B.6}$$

$$k'_7 - k_7 = C_{e_2,k_8,i} + \mathcal{D}_2 - i_2, \tag{B.7}$$

$$k'_8 - k_8 = i - i_1 - (C_{e_1,k'_7,i_2} + \mathcal{D}_1). \tag{B.8}$$

This system gives as a hint how to choose the values of the differences for some of the subkey bits. First note that for each carry produced from a sum of k terms $C_{z_1,\ldots,z_k}$, holds $0 \le C_{z_1,\ldots,z_k} < k$. However, the probability that a carry will take a specific value in this range when z_i are randomly chosen, is not uniformly distributed. For the carries produced from sums with four terms, the probability is highest for the values 1 and 2. Therefore, in our optimized search, we limit the differences $k'_1 - k_1, k_2 - k'_2, k'_5 - k_5, k'_6 - k_6$, to these two values only.

Based on Fig. B.1. the variables $\mathcal{U}_1, \mathcal{U}_2, \mathcal{D}_1, \mathcal{D}_2$, are determined as follows:

$$\mathcal{U}_1 = \big((s_2 + t_2 + C_{s_1,t_1}) \oplus v_2\big) - \big((s_2 + t_2) \oplus v_2\big),$$

$$\mathcal{U}_2 = \big((s_1 + t_1 + C_{s_2,t_2}) \oplus v_2\big) - \big((s_2 + t_2) \oplus v_2\big),$$

$$\mathcal{D}_1 = \big((a_2 + b_2 + C_{a_1,b_1}) \oplus r_2\big) - \big((a_2 + b_2) \oplus r_2\big),$$

$$\mathcal{D}_2 = \big((a_1 + b_1 + C_{a_2,b_2}) \oplus r_1\big) - \big((a_1 + b_1) \oplus r_1\big).$$

It can be checked with a simple computer experiment that these variables can take only odd values and a zero. As C_{w_2,k_4} can be only 0 or 1, it follows that $\mathcal{U}_2$ can take 0, 1, and therefore $k'_3 - k_3$ (see (B.3)) should be checked on the values 1 or 2. A similar reasoning is applicable to the difference $k_4 - k'_4$.

The differences $k'_7 - k_7, k_8 - k'_8$ that are left, are the only one that actually depend on the round counter. However, since $C_{e_2,k_8,i}$ can take the values 0, 1,[3] i.e. it is not fixed but rather flexible, the whole expression $C_{e_2,k_8,i} + \mathcal{D}_2 - i_2$, for any i_2 can take the values 1, 2 (recall that $\mathcal{D}_2$ can be any odd value). Therefore the difference $k'_7 - k_7$ can be 1 or 2 (with probability that depends on the round counter i_2). Finally, let us focus on the difference $k'_8 - k_8$, which is determined by the expression $i - i_1 - C_{e_1,k'_7,i_2} - \mathcal{D}_1$. For a specific counter i, when $k'_7 + e_2 = 0$, the carry C_{e_1,k'_7,i_2} is fixed. Hence in this case, the whole expression can take only one value, 1 or 2, but not both. This limits $k'_8 - k_8$ to only a single value that depends the value of the counter.

[3] It can take the value 2 as well, but the probability is low as the counter i is only 4–5 bits.

The Checksum Subkey Word Recall that k_i, k'_i are the values of the particular subkey words, and not the key words. Out of the five words of the extended key (four master key words and one additional checksum word K_4), in a single subkey addition, four out the these words are chosen. Thus if we fix all of the differences in the four the subkey words of some particular round, then there is only one key differences left to be fixed. If this difference is for the checksum word, then we have to be careful as K_4 is determined from the other words and therefore we would have to fix as well the values (not only differences) of k_1, k_3, k_5, k_7 and the two least significant bits of k_2, k_4, k_6, k_8 so that the difference in K_4 will be as expected. Thus, we fix only two bits because we choose the initial difference to be 1 or 2.

Optimal Values for Bits and Differences Now we are ready to run our optimized search. We look for optimal values (with respect to the rotational probability) for some bits of the first subkey as well as optimal values for the differences between the rotational subkey bits. *The set of possible differences has been reduced by the analysis presented above.* We try all possible values for the two most significant and the two least significant key bits of each subkey word from the first element of the rotational pair of subkey words, and all the corresponding values of the differences from the reduced search set. We keep in mind to comply with the conditions of the checksum word. Finally, to increase the probability we fix the values of the bits 60,61 (the next two bits after the 2 most significant bits) of the first. This results in fixing the two most significant bits of k_2, k_4, k_6, k_8, which in return increases the probability that the carries take the expected values.

We have tested our approach on a real double subkey rounds of Skein-256. That is, the values were found and confirmed to be good by taking rotational input pairs of states and rotational input pair of key words with corrections and testing the probabilities on double subkey rounds. In some cases the theoretical probabilities did not coincide with the empirical. This is due to the fact that there are some hidden dependencies. For example, both $\mathcal{U}_1$ and $k'_5 - k_5$ depend on s_2, t_2.

Initially, each double subkey round was tested independently and the optimal values found were the best for that particular double subkey round, i.e. we found the local maximum. The testing was performed by taking one rotational pair of subkeys with fixed values and differences as described above and around 2^{20} rotational pairs of input states. Then the global maximum was found. We would like to stress out that once this maximum was known, we ran additional experiments, and used the same pair of rotational master key words for finding the probability of all of the double subkey rounds—this ensures to a greater extent that there are no contradictions.

Appendix C. Details on the Acceleration Phase

See tables and figure.

Table C.1. Neutral bits in the acceleration phase. These are used in an inside-out manner, with those computations being eight rounds apart. A single 64-bit word is used, enumeration is from 0 (LSB) to 63 (MSB). The probabilities are measured over 100 right pairs over two rounds backwards and three rounds forwards direction for Skein-256.

bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.
7–17	1.00	18	0.99	19	1.00	20	0.99	21	1.00	22	0.99	23	1.00	24	0.99
25	0.95	26	0.94	27	0.93	28	0.82	31	0.79	33	0.86	36	0.77	38–45	1.00
46	0.99	47	1.00	48	0.99	49	0.98	50	0.97	51	0.96	52	0.96	53	0.96
54	0.90	55	0.84												

Table C.2. Round-by-round rotational probabilities for Skein-256.

Rounds	0–1	2–4	5–8	9–12	13–16	17–20
Prob. $\log_2$	–	−15.13	−21.97	−21.84	−24.44	−24.69
Rounds	21–24	25–28	29–32	33–36	37–40	41–42s
Prob. $\log_2$	−23.83	−26.09	−23.44	−31.75	−27.09	−8.5
Rounds	43–44	45–46	47–48	49–50	51–52	53–54
Prob. $\log_2$	−20.77	−8.5	−20.22	−8.5	−21.53	−8.5

Table C.3. Round-by-round rotational probabilities for Skein-512.

Rounds	0	1–2	3–4	5–6	7–8	9–10	11–12	13–14
Prob. $\log_2$	−13.4	−17.05	−27.88	−17.05	−31.10	−17.05	−23.45	−17.05
Rounds	15–16	17–18	19–20	21–22	23–24	25–26	27–28	29–30
Prob. $\log_2$	−29.15	−17.05	−32.63	−17.05	−30.82	−17.05	−29.73	−17.05
Rounds	31–32	33–34	35–36	37–38	39–40	41–42	43–44	45–46
Prob. $\log_2$	−29.66	−17.05	−29.53	−17.05	−32.60	−17.05	−33.77	−17.05
Rounds	47–48	49–50	51–52	53–54				
Prob. $\log_2$	−30.04	−17.05	−36.76	−17.05				

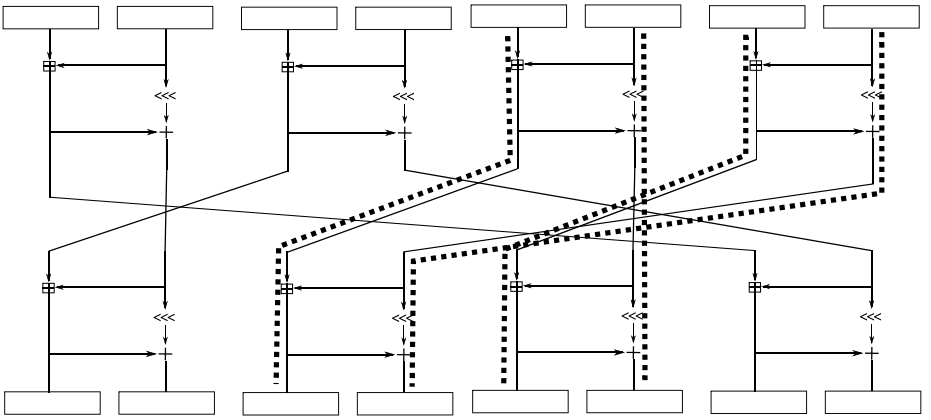

Fig. C.1. Double subkey rounds in Skein-512 divided into two non-overlapping halves.

References

[1] G.V. Assche, A rotational distinguisher on Shabal's keyed permutation and its impact on the security proofs. Available online at http://gva.noekeon.org/papers/ShabalRotation.pdf (2010)

[2] J.-P. Aumasson, Ç. Çalik, W. Meier, O. Özen, R.C.-W. Phan, K. Varici, Improved cryptanalysis of Skein, in *ASIACRYPT'09*. Lecture Notes in Computer Science, vol. 5912 (Springer, Berlin, 2009), pp. 542–559

[3] D.J. Bernstein, Salsa20. Technical Report 2005/025. In *eSTREAM*. ECRYPT Stream Cipher Project (2005). See http://cr.yp.to/snuffle.html

[4] G. Bertoni, J. Daemen, M. Peeters, G.V. Assche, *The Keccak reference*, version **3.0** (2011). See http://keccak.noekeon.org/Keccak-reference-3.0.pdf

[5] E. Biham, R. Chen, Near-collisions of SHA-0, in *CRYPTO'04*. Lecture Notes in Computer Science, vol. 3152 (Springer, Berlin, 2004), pp. 290–305

[6] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, W. Jalby, Collisions of SHA-0 and Reduced SHA-1, in *EUROCRYPT'05*, ed. by R. Cramer. Lecture Notes in Computer Science vol. 3494 (Springer, Berlin, 2005), pp. 36–57

[7] A. Biryukov, D. Khovratovich, I. Nikolić, Distinguisher and related-key attack on the full AES-256, in *CRYPTO'09*. Lecture Notes in Computer Science, vol. 5677 (Springer, Berlin, 2009), pp. 231–249

[8] A. Bogdanov, D. Khovratovich, C. Rechberger, Biclique cryptanalysis of the full AES, in *ASIACRYPT'11*. Lecture Notes in Computer Science, vol. 7073 (Springer, Berlin, 2011), pp. 344–371

[9] J. Chen, K. Jia, Improved related-key boomerang attacks on round-reduced Threefish-512, in *ISPEC*, ed. by J. Kwak, R.H. Deng, Y. Won, G. Wang. Lecture Notes in Computer Science, vol. 6047 (Springer, Berlin, 2010), pp. 1–18

[10] M. Daum, Cryptanalysis of Hash functions of the MD4-family. PhD thesis, Ruhr-Universität Bochum (2005)

[11] A. Duc, J. Guo, T. Peyrin, L. Wei, Unaligned rebound attack: application to Keccak, in *FSE'12*. Lecture Notes in Computer Science, vol. 7549 (Springer, Berlin, 2012), pp. 402–421

[12] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, The Skein hash function family. Submission to NIST. Available at http://www.skein-hash.info/sites/default/files/skein1.1.pdf.

[13] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, Provable security support for the Skein hash family (2009). Available at www.skein-hash.info/sites/default/files/skein-proofs.pdf

[14] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, The Skein hash function family, version 1.2 (2009). Submission to NIST (Round 2), Available at http://www.skein-hash.info/sites/default/files/skein1.2.pdf

[15] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, The Skein hash function family, version 1.3 (2010). Submission to NIST (Round 3). Available at http://www.skein-hash.info/sites/default/files/skein1.3.pdf

[16] S. Indesteege, F. Mendel, B. Preneel, C. Rechberger, Collisions and other non-random properties for step-reduced SHA-256, in *Selected Areas in Cryptography'08*, ed. by R.M. Avanzi, L. Keliher, F. Sica. Lecture Notes in Computer Science, vol. 5381 (Springer, Berlin, 2008), pp. 276–293

[17] J. Jean, M. Naya-Plasencia, T. Peyrin, Improved rebound attack on the finalist Grøstl, in *FSE'12*. Lecture Notes in Computer Science, vol. 7549 (Springer, Berlin, 2012), pp. 110–126

[18] J. Jean, M. Naya-Plasencia, M. Schläffer, Improved analysis of ECHO-256, in *Selected Areas in Cryptography'11*. Lecture Notes in Computer Science, vol. 7118 (Springer, Berlin, 2011), pp. 19–36

[19] A. Joux, T. Peyrin, Hash functions and the (amplified) boomerang attack, in *CRYPTO'07*. Lecture Notes in Computer Science, vol. 4622 (Springer, Berlin, 2007), pp. 244–263

[20] D. Khovratovich, Bicliques for permutations: collision and preimage attacks in stronger settings, in *ASIACRYPT'12*. Lecture Notes in Computer Science, vol. 7658 (Springer, Berlin, 2012), pp. 544–561

[21] D. Khovratovich, M. Naya-Plasencia, A. Röck, M. Schläffer, Cryptanalysis of Luffa v2 components, in *Selected Areas in Cryptography'10*. Lecture Notes in Computer Science, vol. 6544 (Springer, Berlin, 2010), pp. 388–409

[22] D. Khovratovich, I. Nikolić, Rotational cryptanalysis of ARX, in *FSE'10*. Lecture Notes in Computer Science, vol. 6147 (Springer, Berlin, 2010), pp. 333–346

[23] D. Khovratovich, I. Nikolić, C. Rechberger, Rotational rebound attacks on reduced Skein, in *ASIACRYPT'10*. Lecture Notes in Computer Science, vol. 6477 (Springer, Berlin, 2010), pp. 1–19

[24] D. Khovratovich, C. Rechberger, A. Savelieva, Bicliques for preimages: attacks on Skein-512 and the SHA-2 family, in *FSE'12*. Lecture Notes in Computer Science, vol. 7549 (Springer, Berlin, 2012), pp. 244–263

[25] V. Klima, Tunnels in hash functions: MD5 collisions within a minute (2006). Available at http://eprint.iacr.org/2006/105.pdf

[26] M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, M. Schläffer, Rebound distinguishers: results on the full Whirlpool compression function, in *ASIACRYPT'09*. Lecture Notes in Computer Science, vol. 5912 (Springer, Berlin, 2009), pp. 126–143

[27] G. Leurent, Construction of differential characteristics in ARX designs—application to Skein. *Cryptology*. ePrint Archive, Report 2012/668 (2012)

[28] G. Leurent, A. Roy, Boomerang attacks on Hash function using auxiliary differentials, in *CT-RSA'12*. Lecture Notes in Computer Science, vol. 7178 (Springer, Berlin, 2012), pp. 215–230

[29] J. Li, T. Isobe, K. Shibutani, Converting meet-in-the-middle preimage attack into pseudo collision attack: application to SHA-2, in *FSE'12*. Lecture Notes in Computer Science, vol. 7549 (Springer, Berlin, 2012), pp. 264–286

[30] K. Matusiewicz, M. Naya-Plasencia, I. Nikolić, Y. Sasaki, M. Schläffer, Rebound attack on the full LANE compression function, in *ASIACRYPT'09*. Lecture Notes in Computer Science, vol. 5912 (Springer, Berlin, 2009), pp. 106–125

[31] F. Mendel, T. Nad, M. Schläffer, Finding SHA-2 characteristics: searching through a minefield of contradictions, in *ASIACRYPT'11*. Lecture Notes in Computer Science, vol. 7073 (Springer, Berlin, 2011), pp. 288–307

[32] F. Mendel, T. Peyrin, C. Rechberger, M. Schläffer, Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher, in *Selected Areas in Cryptography'09*. Lecture Notes in Computer Science, vol. 5867 (Springer, Berlin, 2009), pp. 16–35

[33] F. Mendel, N. Pramstaller, C. Rechberger, V. Rijmen, Analysis of step-reduced SHA-256, in *FSE'06*. Lecture Notes in Computer Science, vol. 4047 (Springer, Berlin, 2006), pp. 126–143

[34] F. Mendel, C. Rechberger, M. Schläffer, S.S. Thomsen, The rebound attack: cryptanalysis of reduced Whirlpool and Grøstl, in *FSE'09*. Lecture Notes in Computer Science, vol. 5665 (Springer, Berlin, 2009), pp. 260–276

[35] P. Morawiecki, J. Pieprzyk, M. Srebrny, Rotational cryptanalysis of round-reduced Keccak. *Cryptology*. ePrint Archive, Report 2012/546 (2012). http://eprint.iacr.org/

[36] Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro, K. Ohta, Improved collision search for SHA-0, in *ASIACRYPT'06*. Lecture Notes in Computer Science, vol. 4284 (Springer, Berlin, 2006), pp. 21–36

[37] I. Nikolić, A. Biryukov, Collisions for step-reduced SHA-256, in *FSE'08*. Lecture Notes in Computer Science, vol. 5086 (Springer, Berlin, 2008), pp. 1–15

[38] I. Nikolić, J. Pieprzyk, P. Sokolowski, R. Steinfeld, Rotational cryptanalysis of (modified) versions of BMW and SIMD (2010). Available online at https://cryptolux.org/mediawiki/uploads/0/07/Rotational_distinguishers_(Nikolic,_Pieprzyk,_Sokolowski,_Steinfeld).pdf

[39] V. Rijmen, E. Oswald, Update on SHA-1, in *CT-RSA*, ed. by A. Menezes. Lecture Notes in Computer Science, vol. 3376 (Springer, Berlin, 2005), pp. 58–71

[40] Y. Sasaki, K. Yasuda, Known-key distinguishers for 11-round Feistel ciphers: application to collision attacks on their hashing modes, in *FSE'11*. Lecture Notes in Computer Science, vol. 6733 (Springer, Berlin, 2011), pp. 397–415

[41] F.-X. Standaert, G. Piret, N. Gershenfeld, J.-J. Quisquater, SEA: a scalable encryption algorithm for small embedded applications, in *CARDIS'06*. Lecture Notes in Computer Science, vol. 3928 (Springer, Berlin, 2006), pp. 222–236

[42] M. Stevens, On collisions for MD5. Master's thesis, Eindhoven University of Technology, Eindhoven, Netherlands (2007)

[43] B. Su, W. Wu, S. Wu, L. Dong, Near-collisions on the reduced-round compression functions of Skein and BLAKE, in *CANS'10*. LNCS, vol. 6467 (Springer, Berlin, 2010), pp. 124–139

[44] X. Wang, Y.L. Yin, H. Yu, Finding collisions in the full SHA-1, in *CRYPTO*, ed. by V. Shoup. Lecture Notes in Computer Science, vol. 3621 (Springer, Berlin, 2005), pp. 17–36

[45] X. Wang, H. Yu, How to break MD5 and other hash functions, in *EUROCRYPT'05*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 19–35
[46] S. Wu, D. Feng, W. Wu, Practical rebound attack on 12-round Cheetah-256, in *ICISC*, ed. by D. Lee, S. Hong. Lecture Notes in Computer Science, vol. 5984 (Springer, Berlin, 2009), pp. 300–314
[47] H. Yu, J. Chen, X. Wang, Partial-collision attack on the round-reduced compression function of Skein-256, in FSE'13 (2013)