Improved Cryptanalysis of Skein

Jean-Philippe Aumasson[1,*], Çağdaş Çalık[2], Willi Meier[1,†], Onur Özen[3], Raphael C.-W. Phan[4], and Kerem Varıcı[5,‡]

[1] FHNW, Klosterzelgstrasse 2, 5210 Windisch, Switzerland
[2] Middle East Technical University, Institute of Applied Mathematics, 06531 Ankara, Turkey
[3] EPFL IC LACAL Station 14, 1015 Lausanne, Switzerland
[4] Loughborough Uni, Electronic and Electrical Engineering, LE11 3TU, UK
[5] K.U.Leuven, Dept. of Electrical Engineering, ESAT/SCD/COSIC and IBBT Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

Abstract. The hash function Skein is the submission of Ferguson et al. to the NIST Hash Competition, and is arguably a serious candidate for selection as SHA-3. This paper presents the first third-party analysis of Skein, with an extensive study of its main component: the block cipher Threefish. We notably investigate near collisions, distinguishers, impossible differentials, key recovery using related-key differential and boomerang attacks. In particular, we present near collisions on up to 17 rounds, an impossible differential on 21 rounds, a related-key boomerang distinguisher on 34 rounds, a known-related-key boomerang distinguisher on 35 rounds, and key recovery attacks on up to 32 rounds, out of 72 in total for Threefish-512. None of our attacks directly extends to the full Skein hash. However, the pseudorandomness of Threefish is required to validate the security proofs on Skein, and our results conclude that at least 36 rounds of Threefish seem required for optimal security guarantees.

1 Introduction

The hash function research scene has seen a surge of works since devastating attacks [1–4] on the two most deployed hash functions, MD5 and SHA-1. This led to a lack of confidence in the current U.S. (and de facto worldwide) hash standard, SHA-2 [5], because of its similarity with MD5 and SHA-1.

As a response to the potential risks of using SHA-2, the U.S. National Institute of Standards and Technology (NIST) launched a public competition—the

[*]Supported by the Swiss National Science Foundation under project no. 113329.

[†]Supported by GEBERT RÜF STIFTUNG, project no. GRS-069/07.

[‡]This work was sponsored by the Research Fund K.U.Leuven, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT Programme under Contract ICT-2007-216676 (ECRYPT II). The information in this paper is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

NIST Hash Competition—to select a new hash standard [6]. The new hash function, SHA-3, is expected to have at least the security of SHA-2, and to achieve this with significantly improved efficiency. By the deadline of October 2008, NIST received 64 submissions, 51 were accepted as first round candidates, and in July 2009 14 were selected as second round candidates, including Skein. Due to the critical role of hash functions in security protocols, this competition catches the attention not only from academia, but also from industry—with candidates from IBM, Hitachi, Intel, Sony—and from governmental organizations.

Skein [7] is the submission of Ferguson et al. to the NIST Hash Competition. According to its designers, it combines "speed, security, simplicity and a great deal of flexibility in a modular package that is easy to analyze" [7, p.i]. Skein supports three different internal state sizes (256-, 512-, and 1024-bit), and is one of the fastest contestants on 64-bit machines.

Skein is based on the "UBI (The Unique Block Iteration) chaining mode" that itself uses a compression function made out of the Threefish-512 block cipher. Below we give a brief top-down description of these components:

- **Skein** makes three invocations to the UBI mode with different *tags*: the first hashes the configuration block with a tag "Cfg", the second hashes the message with a tag "Msg", and the third hashes a null value with a tag "Out".
- **UBI** mode hashes an arbitrary-length string by iterating invocations to a compression function, which takes as input a chaining value, a message block, and a tweak. The tweak encodes the number of bytes processed so far, and special flags for the first and the last block.
- **The compression function** inside the UBI mode is the Threefish-512 block cipher in MMO (Matyas-Meyer-Oseas) mode, i.e., from a chaining value h, a message block m, and a tweak t it returns $E_h(t,m) \oplus m$ as new chaining value.
- **Threefish** is a family of tweakable block ciphers based on a simple permutation of two 64-bit words: $\mathrm{MIX}(x,y) = (x+y, (x+y) \oplus (y \lll R))$. Threefish-512 is the version of Threefish with 512-bit key and 512-bit blocks, and is used in the default version of Skein.

So far, no third-party cryptanalysis of Skein has been published, and the only cryptanalytic results are in its documentation [7, §9]. It describes a near collision on eight rounds for the compression function, a distinguisher for 17 rounds of Threefish, and it conjectures the existence of key recovery attacks on 24 to 27 rounds (depending on the internal state size). Furthermore, [7, §9] discusses the possibility of a trivial related-key boomerang attack on a modified Threefish, and concludes that it cannot work on the original version. A separate document [8] presents proofs of security for Skein when assuming that some of its components behave ideally (e.g., that Threefish is an ideal cipher).

This paper presents the first external analysis of Skein, with a focus on the main component of its default version: the block cipher Threefish-512. Table 1 summarizes our results.

Table 1. Summary of the known results on Threefish-512 (near collisions are for Threefish-512 in MMO mode, related-key boomerang attacks make use of four related-keys ,"$\surd$" designates the present paper).

Rounds	Time	Memory	Type	Authors
8	1	–	511-bit near-collision	[7]
16	2^{6}	–	459-bit near-collision	$\surd$
17	2^{24}	–	434-bit near-collision	$\surd$
17	$2^{8.6}$	–	related-key distinguisher*	[7]
21	$2^{3.4}$	–	related-key distinguisher	$\surd$
21	–	–	related-key impossible differential	$\surd$
25	?	–	related-key key recovery (conjectured)	[7]
25	$2^{416.6}$	–	related-key key recovery	$\surd$
26	$2^{507.8}$	–	related-key key recovery	$\surd$
32	2^{312}	2^{71}	related-key boomerang key recovery	$\surd$
34	2^{398}	–	related-key boomerang distinguisher	$\surd$
35	2^{478}	–	known-related-key boomerang distinguisher	$\surd$

$\star$: complexity deduced from the biases in [7, Tab.22].

The rest of the paper is organized as follows: §2 describes Threefish-512; §3 studies near-collisions for Skein's compression function with a reduced Threefish-512; §4 describes impossible differentials; §5 discusses and improves the key-recovery attacks sketched in [7, §§9.3]. Finally, §6 uses the boomerang technique to describe our best distinguishers and key-recovery attacks on Threefish. §7 concludes.

2 Brief Description of Threefish-512

Threefish-512 works on 64-bit words, and we write their hexadecimal value in sans-serif font (e.g., 0123456789ABCDEF). The letter Δ stands for a difference in the most significant bit (MSB), i.e., $\Delta =$ 8000000000000000. Notations are the same as in the specification of Threefish [7, §§2.2]: a 512-bit plaintext block is parsed as eight words $v_{0,0}, \dots, v_{0,7}$, and is encrypted through $N_r = 72$ rounds, where round number $d \in \{0, \dots, N_r - 1\}$ operates as follows:

1. If $d \equiv 0 \bmod 4$, add a subkey by setting $e_{d,i} \leftarrow v_{d,i} + k_{d,i}$, $i = 0, \dots, 7$, otherwise, just copy the state $e_{d,i} \leftarrow v_{d,i}$, $i = 0, \dots, 7$.
2. Set $(f_{d,2i}, f_{d,2i+1}) \leftarrow \mathrm{MIX}_{d,i}(e_{d,2i}, e_{d,2i+1})$, $i = 0, \dots, 3$, where
$$\mathrm{MIX}_{d,i}(x, y) = (x + y, (x + y) \oplus (y \lll R_{d,i})) \ ,$$
with $R_{d,i}$ a rotation constant dependent on d and i.
3. Permute the state words:
$$\begin{array}{llll} v_{d+1,0} \leftarrow f_{d,2} & v_{d+1,1} \leftarrow f_{d,1} & v_{d+1,2} \leftarrow f_{d,4} & v_{d+1,3} \leftarrow f_{d,7} \\ v_{d+1,4} \leftarrow f_{d,6} & v_{d+1,5} \leftarrow f_{d,5} & v_{d+1,6} \leftarrow f_{d,0} & v_{d+1,7} \leftarrow f_{d,3} \ . \end{array}$$

After $N_r \equiv 0 \bmod 4$ rounds, the ciphertext is set to

$$(v_{N_r,0} + k_{N_r,0}), \ldots, (v_{N_r,7} + k_{N_r,7}) \ .$$

The s-th keying (counting from zero, thus which occurs at round $d = 4s$) uses subkeys $k_{s,0}, \ldots, k_{s,7}$. These are derived from the key $k_0, \ldots, k_7$ and from the tweak t_0, t_1 as

$$\begin{array}{l|l}
k_{s,0} \leftarrow k_{(s+0) \bmod 5} & k_{s,4} \leftarrow k_{(s+4) \bmod 5} \\
k_{s,1} \leftarrow k_{(s+1) \bmod 5} & k_{s,5} \leftarrow k_{(s+5) \bmod 5} + t_{s \bmod 3} \\
k_{s,2} \leftarrow k_{(s+2) \bmod 5} & k_{s,6} \leftarrow k_{(s+6) \bmod 5} + t_{(s+1) \bmod 3} \\
k_{s,3} \leftarrow k_{(s+3) \bmod 5} & k_{s,7} \leftarrow k_{(s+7) \bmod 5} + s
\end{array}$$

where $k_8 = \texttt{5555555555555555} \oplus \bigoplus_{i=0}^{7} k_i$ and $t_2 = t_0 \oplus t_1$.

3 Near Collisions for the UBI Compression Function

We extend the analysis presented in [7, §9] to find near-collisions for the compression function of Skein's UBI mode; [7, §9] exploits *local collisions*, i.e., collisions in the intermediate values of the state, which occur when particular differences are set in the key, the plaintext, and the tweak.

The compression function outputs $E_k(t, m) \oplus m$, where E is Threefish-512. Our strategy is simple: like in [7, §9], we prepend a four-round differential trail to the first local collision at round four so as to avoid differences until the 13-th round. Then, we follow the trail induced by the introduced difference.

The next two sections work out the details as follows:

- §§3.1 shows how to adapt the differential trail found in [7, §9] when a 4-round trail is prepended.
- §§3.2 describes the differential trails used and evaluates the probability that a random input conforms.
- §§3.3 explains how to reduce the complexity of the attack by precomputing a single conforming pair for the first 4-round trail, and using some conditions to speed up the search.

3.1 Adapting Differences in the Key and the Tweak

In [7, §§§9.3.4], Skein's designers suggest to prepend a 4-round trail that leads to the difference $(0, 0, \ldots, 0, \Delta)$, previously used for the 8-round collision. However, the technique as it is presented does not work. This is because the order of keyings is then shifted, and so the original difference in the key and in the tweak does not cancel the $(0, 0, \ldots, 0, \Delta)$ difference at the second keying.

Therefore, for differences to vanish at the third keying, one needs a difference Δ in k_7 and t_0, which gives a difference $(0, \ldots, 0, \Delta)$ at the second keying, and $(0, 0, 0, 0, \Delta, 0, 0)$ after the fourth. The difference in the state after $(4+8)$ rounds is thus the same as originally after eight rounds. Note that, as observed in [7, §§9.4], at least seven keyings separate two vanishing keyings. See Table 2 for details.

Table 2. Details of the subkeys and of their differences, given a difference Δ in k_7 and t_0 (leading to Δ differences in k_8 and t_2).

s	d	$k_{s,0}$	$k_{s,1}$	$k_{s,2}$	$k_{s,3}$	$k_{s,4}$	$k_{s,5}$	$k_{s,6}$	$k_{s,7}$
		Differences							
0	0	k_0	k_1	k_2	k_3	k_4	$k_5 + t_0$	$k_6 + t_1$	k_7
		0	0	0	0	0	Δ	0	Δ
1	4	k_1	k_2	k_3	k_4	k_5	$k_6 + t_1$	$k_7 + t_2$	$k_8 + 1$
		0	0	0	0	0	0	0	Δ
2	8	k_2	k_3	k_4	k_5	k_6	$k_7 + t_2$	$k_8 + t_0$	$k_0 + 2$
		0	0	0	0	0	0	0	0
3	12	k_3	k_4	k_5	k_6	k_7	$k_8 + t_0$	$k_0 + t_1$	$k_1 + 3$
		0	0	0	0	Δ	0	0	0
4	16	k_4	k_5	k_6	k_7	k_8	$k_0 + t_1$	$k_1 + t_2$	$k_2 + 4$
		0	0	0	Δ	Δ	0	Δ	0
5	20	k_5	k_6	k_7	k_8	k_0	$k_1 + t_2$	$k_2 + t_0$	$k_3 + 5$
		0	0	Δ	Δ	0	Δ	Δ	0
6	24	k_6	k_7	k_8	k_0	k_1	$k_2 + t_0$	$k_3 + t_1$	$k_4 + 6$
		0	Δ	Δ	0	0	Δ	0	0

3.2 Differential Trails

We now trace the difference when prepending four rounds, i.e., when the difference is in k_7 and in t_0 only (and in the plaintext).

4-Round Trail. To prepend four rounds and reach the difference $(0, \ldots, 0, \Delta)$, one uses the trail provided in the full version [9] of this paper. The plaintext difference is modified by the first keying (the MSB differences in the sixth and eighth word vanish). The probability that a random input successfully crosses the 4-round differential trail is 2^{-33} (either forward or backward).

12-Round Trail. The second keying adds Δ to the last state word, making its difference vanish. The state remains free of any difference up to the fourth keying, after the twelfth round, which sets a difference Δ in the fifth word state. Table 3 presents the corresponding trail for up to the 17-th round. After 17 rounds, the weight becomes too large to obtain near collisions. On 16 rounds, adding the final keying and the feedforward, one obtains a collision on $512 - 53 = 459$ bits. Likewise, for 17 rounds, a collision can be found on $512 - 78 = 434$ bits.

3.3 Optimizing the Search

A direct application of the differential trails in the previous section gives a cost 2^{33} to cross the first four rounds; then, after the twelfth round,

Table 3. Differential trail (linearization) used for near collisions, of probability 2^{-24}.

Rd	Difference	Pr
13	`0000000000000000 0000000000000000 8000000000000000 0000000000000000`	1
	`0000000000000000 8000000000000000 0000000000000000 0000000000000000`	
14	`8000000000000000 0000000000000000 8000000000000000 0000000000000000`	1
	`0000000000000000 8000010000000000 0000000000000000 8000000000000000`	
15	`8000000000000000 8000000000000000 8000010000000000 8000000000000100`	2^{-1}
	`8000000000000000 8008010000000400 8000000000000000 8000000000000000`	
16	`0000010000000100 0000000100000000 0008010000000400 0000000400000000`	2^{-5}
	`0000000000000000 000A014004008400 0000000000000000 0804010000000100`	
17	`8008010400000400 0000010100000140 800A014004008400 A805018020000100`	2^{-18}
	`8804010000000100 900A016801009402 0000010100000100 8008010420000401`	

- With 16 rounds: complexity is $2^{1+5} = 2^6$, so 2^{39} in total, for finding a collision over 459 bits.
- With 17 rounds: complexity is $2^{1+5+18} = 2^{24}$, so 2^{56} in total, for finding a collision over 434 bits.

A simple trick allows us to avoid the cost of crossing the first 4-round trail: note that the first keying adds $(k_5 + t_0)$ to the sixth state word, and $(k_6 + t_1)$ to the seventh; hence, given one conforming pair, one can modify k_5, k_6, t_0, t_1 while preserving the values of $(k_5 + t_0)$ and $(k_6 + t_1)$, and the new input will also follow the differential trail. It is thus sufficient to precompute a single conforming pair to avoid the cost due to the prepended rounds.

To carry out this precomputation efficiently, a considerable speedup of the 2^{33} complexity can be obtained by finding sufficient conditions to cross the first round with probability one (instead of 2^{-21}):

- A first set of conditions is on the words (v_{2i}, v_{2i+1}): whenever there is a nonzero difference at a same offset, the bit should have a different value in the first and in the second word (otherwise carries induce additional differences).
- A second set of conditions concerns the differences that do not "collide": one should ensure that no carry propagates from the leftmost bits.

In total, there are $13 + 8 = 21$ such conditions, which lets enough degrees of freedom to satisfy the subsequent differential tails. Using techniques like neutral bits [10], the probability may be reduced further, but the complexity 2^{12} is low enough for efficiently finding a conforming pair. By choosing inputs according to the above conditions, while being careful to avoid contradictions, we can find a pair that conforms within a few thousand trials (see Appendix A for an example).

We can now use this pair to search for near collisions. It suffices to pick random values for k_5 and k_6, then set $t_0 = -k_5$ and $t_1 = -k_6$ to get a set of 2^{128} distinct inputs. Experiments were consistent with our analysis, and examples of near collisions are given in Appendix B.

3.4 Improved Distinguisher

Based on our trick to cross the first twelve rounds "for free", we can improve the distinguisher suggested in [7]. This distinguisher exploited the observation of a bias $0.01 < \varepsilon \leq 0.05$ after 17 rounds (thus leading to a distinguisher requiring at least $1/0.05^2 \approx 400$ samples). [7] suggested to combine it with the prepending of four rounds, though no further details were given. Our observations show that with the adapted difference in the key and the tweak, a bias about 0.3 exists at the 385-th bit, after 21 rounds. We detected this bias using a frequency test similar to that in [11, §§2.1]. This directly gives a distinguisher on 21 rounds, and requiring only about $1/0.3^2 \approx 11$ samples.

4 Impossible Differentials

The *miss-in-the-middle* technique (a term coined by Biham et al. in [12]), was first applied by Knudsen [13] to construct a 5-round impossible differential for the block cipher DEAL. The idea was generalized by Biham et al. [12] to find impossible differentials for ciphers of any structure. The idea is as follows: Consider a cascade cipher $E = E^\beta \circ E^\alpha$ such that for E^α there exists a differential $(\varDelta^\alpha_{\text{in}} \rightarrow \varDelta^\alpha_{\text{out}})$ and for $(E^\beta)^{-1}$ there exists a differential $(\varDelta^\beta_{\text{in}} \rightarrow \varDelta^\beta_{\text{out}})$, both with *probability one*, where the equality is impossible $(\varDelta^\alpha_{\text{out}} \neq \varDelta^\beta_{\text{out}})$. It follows that the differential $(\varDelta^\alpha_{\text{in}} \rightarrow \varDelta^\beta_{\text{in}})$ cannot occur, for it requires $\varDelta^\alpha_{\text{out}} = \varDelta^\beta_{\text{out}}$. This technique can be extended to the related-key setting. For example, related-key impossible differentials were found for 8-round AES-192 [14, 15].

Below we first present probability-1 truncated differentials on the first 13 rounds (forward) and on the last seven rounds (backward) of 20-round Threefish-512. A "miss-in-the-middle" observation then allows us to deduce the existence of impossible differentials on 20 and 21 rounds.

4.1 Forward Differential

The first keying ($s = 0$) adds to the state $v_{0,i}, \ldots, v_{0,7}$ the values $k_0, k_1, \ldots, k_4, k_5 + t_0, k_6 + t_1, k_7$. Then, the second keying ($s = 1$) adds $k_1, \ldots, k_5, k_6 + t_1, k_7 + t_2, k_8 + 1$. By setting a difference $\varDelta$ in k_6, k_7, t_1 and in the plaintext $v_{0,7}$, we ensure that differences vanish in the first two keyings, and thus nonzero differences only appear after the eighth round, for third keying.

The third keying ($s = 2$) adds $k_2, \ldots, k_6, k_7 + t_2, k_8 + t_0, k_0 + t_2$. Hence the difference $\varDelta$ is introduced in $e_{8,4}$ only. It gives a difference $\varDelta$ in $f_{8,4}, f_{8,5}$, thus in $v_{9,2}, v_{9,5}$. After the tenth round, the state $v_{10,\cdot}$ has the following difference with probability one.

```
8000000000000000  0000000000000000  8000000000000000  0000000000000000
0000000000000000  8000040000000000  0000000000000000  8000000000000000 .
```

After the twelfth round (before the fourth keying), the state $v_{12,\cdot}$ has again some differences that occur with probability one (the `X` differences are uncertain, that is, have probability strictly below one):

```
XXXXXXXXX4000000  0000000002000000  XXXXXXXXXXXX4000  0000000000000040
0000000000000000  XXXXXXXXXXXXX100  0000000000000000  XXXXXXXXX4000800 .
```

Given this class of differences, after the 13-th round (which starts by making the fourth keying) we have the class of differences

```
XXXXXXXXXXXXXX40  XXXXXXXXX2000000  XXXXXXXXXXXXX100  XXXXXXXXXXXXXX10
XXXXXXXXXXXXX800  XXXXXXXXXXXXXXXX  XXXXXXXXX2000000  XXXXXXXXXXXXXX40 .
```

There are in total 92 bits with probability-1 differences between the 13-th and the 14-th round. These differences were empirically verified.

4.2 Backward Differential

The sixth keying ($s = 5$), which occurs after the 20-th round, returns the ciphertext

$$\begin{array}{rl|rl} c_0 &= v_{20,0} + k_5 & c_4 &= v_{20,4} + k_0 \\ c_1 &= v_{20,1} + k_6 & c_5 &= v_{20,5} + k_1 + t_2 \\ c_2 &= v_{20,2} + k_7 & c_6 &= v_{20,6} + k_2 + t_0 \\ c_3 &= v_{20,3} + k_8 & c_7 &= v_{20,7} + k_3 + 5 \end{array}$$

By setting a difference Δ in k_6, k_7, t_1 (like for the forward differential), and in the ciphertext words c_1, c_2, c_5, we ensure that differences vanish in the sixth keying, and thus nonzero differences only appear after the 17-th round, when making the fifth keying (by computing backwards from the 20-th round).

The fifth keying ($s = 4$), after the 16-th round, subtracts from the state the values $k_4, \ldots, k_8, k_0 + t_1, k_1 + t_2, k_2 + 4$. Hence, the difference Δ is introduced (backwards) in $v_{16,2}, v_{16,3}, v_{16,5}, v_{16,6}$. After inverting the 16-th round, we obtain with probability one the difference

```
XXXXXXXX40000000  0000000040000000  0000000000000000  0000000000000000
8000000000000000  0000000000000000  XXXXXXXX10000000  0000000010000000 .
```

Finally, after inverting the 14-th round, we have the following difference with probability one:

```
XXXXXXXXXXXX8000  XXXXXXXXXXXX8000  XXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX  XXXXXXXXXX400000  XXXXXXXXXX800000  XX50000000800000 .
```

In total there are 134 bits of difference with probability one between the 14-th and the 13-th round.

4.3 Miss-in-the-Middle

We showed that if there's a difference Δ in the key in k_6 and k_7, and in the tweak in t_1, then a difference Δ in the plaintext word $v_{0,7}$ propagates to give probability-1 differences after up to 13 rounds. Then we showed that for the same difference in the key and in the tweak, a difference Δ in the ciphertext words c_1, c_2, c_5 guarantees (probability one) that between the 13-th and the 14-th rounds we also have probability-1 differences.

Looking for example at the first word of the state: the forward differential leads to a difference in the seventh bit, whereas the backward differential requires this bit to be unchanged. Therefore, it is impossible that a difference Δ in the plaintext $v_{0,7}$ leads to a difference Δ in c_1, c_2, c_5 with 20-round Threefish-512.

We can extend this impossible differential one more round: after the 20-th round and the sixth keying the state has only differences Δ in $e_{20,1}, e_{20,2}, e_{20,3}$. These differences always give the same difference after the 21-st round, because they are only in MSB's. This directly gives an impossible differential on 21 rounds of Threefish-512 (e.g., 21 out of 72). However contrary to the 20-round impossible differential, it is irrelevant to Threefish-512 with exactly $N_r = 21$ rounds, because of the final keying that occurs after the 21-st round (which makes some differences uncertain, because before the keying we have differences in non-MSB's).

5 Improved Key-Recovery Attacks

The documentation of Skein sketches key-recovery attacks on all Threefish versions, though the complexity is not studied. We analyzed these observations, and could find better attacks than conjectured by the Skein designers.

To optimize the attack strategy in [7, §§9.3], the attacker has to determine which key bits should be guessed. This is to minimize the noise over the bias after a partial inversion of the last rounds, and thus to minimize the complexity of the attack. The less key bits guessed, the better for the attacker (up to the bound of half the key bits). One can easily determine which key bits do not affect the bias when inverting one or two rounds. For example, two rounds after round 21 (where the bias occurs), the 385-th bit does not affect the second, third, fourth, and sixth state words. Hence, it is not affected by a wrong guess of the key words k_0, k_2, k_6. The bias is slightly affected by erroneous guesses of k_3 (which modifies the last state word in the keying), but it is still large (about $0.12 \approx 2^{-3}$). It is thus sufficient to guess half the key (k_1, k_4, k_5, k_7) to be able to observe the bias.

Note that the cost of the prepended rounds depends on which key words are guessed: indeed, when guessing a word, one can adapt the corresponding plaintext word in order to satisfy the conditions of the differential. Here the non-guessed words imply a cost $2^{12+18} = 2^{30}$ to cross the first differential. The total cost of recovering the 512-bit key on 23 rounds is thus about $2^{30} \times 2^{6} \times 2^{256} = 2^{292}$.

To attack more rounds, a more advanced search for the optimal set of bits to be guessed is likely to reduce the complexity of our attacks. For this, we used the same strategy as in the analysis of the Salsa20 and ChaCha stream ciphers [16]. Namely, we computed the *neutrality* of each key bit (i.e., the probability that flipping the bit preserves the difference), and we chose to guess the bits that affect the bias the most, using some threshold on their neutrality. More precisely, we sort key bits according to their neutrality, then filter them with respect to some threshold value. According to [16]'s terminology, this corresponds to partitioning the key bits into "significant" and "non-significant" ones.

Recall that in §§3.4 we observed a bias at the 385-th bit after $4 + 17$ rounds of Threefish-512. A key recovery attack on $21 + n$ rounds consists in guessing some key bits, inverting n rounds based on this guess, letting the other key bits be random, and observing a bias in that bit. Complexity is determined by the number of guessed bits and the value of the observed bias.

Inverting four rounds with all key bits whose neutrality is greater than 0.29 (we found 125 of those), we observe a bias 0.0365. Since some key bits are not guessed, and thus assumed random, some of the conditions to conform to the first round's differential cannot be controlled. There are eight such additional conditions, which means that the 4-round initial differential will be followed with probability 2^{-12-8}. Since our bias approximately equals to $2^{-4.8}$, and since we need to guess $512 - 125$ key bits, the overall complexity of the attack on 25-round Threefish-512 is about $2^{12+8} \times 2^{2\times 4.8} \times 2^{387} = 2^{416.6}$. Below we give the mask corresponding to the 125 *non-guessed* bits, for each key word:

```
0000070060FFF836  0040030021FFFC0E  803C02F03FFFF83F  001001001603C006
00780E30007F000E  0000000000000000  0000000000000000  007001800E03F801 .
```

We can apply the same method on 26 rounds: with a neutrality threshold 0.17 we obtain 30 "significant" key bits, and we observe a bias about 0.017 when all of them are random. The non-guessed bits give two additional conditions for the first 4-round differential. In total, the complexity of the attack is thus about $2^{12+2} \times 2^{2\times 5.9} \times 2^{482} = 2^{507.8}$. Memory requirements are negligible.

6 Boomerang Attacks

Boomerang attacks were introduced by Wagner and first applied to block ciphers [17]. Roughly speaking, in boomerang attacks one uses two short differential trails rather than a long one to exploit the efficiency of the former trails. Let E denote the encryption function of Threefish. View E as a cascade of four subciphers

$$E = E^{\omega} \circ E^{\gamma} \circ E^{\beta} \circ E^{\alpha} \ , \tag{1}$$

so that E is composed of a core $E' = E^{\gamma} \circ E^{\beta}$ sandwiched by rounds E^{α} and E^{ω}. The boomerang distinguisher is generally described for E' only, but for key recovery attacks on Threefish we need to generalize the attack to the construction in Eq. (1).

Recall that in related-key attacks, one assumes that the attacker can query the cipher with other keys that have some specified relation with the original key. This relation is often an XOR-difference. A related-key differential is thus a triplet $(\Delta_{\text{in}}, \Delta_{\text{out}}, \Delta_k)$, associated with the probability

$$\Pr_{k,m}\left[E_k(m) \oplus E_{k\oplus\Delta_k}(m \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}\right] = p \ .$$

Here, Δ_{in} and Δ_{out} are the input and output differences, Δ_k is the key difference, and p the probability of the differential.

For (related-key) boomerang attacks based on four related-keys, one exploits two short related-key differentials: $(\Delta^\beta_{\text{in}}, \Delta^\beta_{\text{out}}, \Delta^\beta_k)$ for E^β, of probability p and $(\Delta^\gamma_{\text{in}}, \Delta^\gamma_{\text{out}}, \Delta^\gamma_k)$ for E^γ, of probability q.

A distinguisher then works as follows:

1. Pick a random plaintext m_1 and form $m_2 = m_1 \oplus \Delta^\beta_{\text{in}}$.
2. Obtain $c_1 = E'_k(m_1)$ and $c_2 = E'_{k\oplus\Delta^\beta_k}(m_2)$.
3. Set $c_3 = c_1 \oplus \Delta^\gamma_{\text{out}}$ and $c_4 = c_2 \oplus \Delta^\gamma_{\text{out}}$.
4. Obtain $m_3 = E'^{-1}_{k\oplus\Delta^\gamma_k}(c_3)$ and $m_4 = E'^{-1}_{k\oplus\Delta^\beta_k\oplus\Delta^\gamma_k}(c_4)$.
5. Check $m_3 \oplus m_4 = \Delta^\beta_{\text{in}}$.

For an ideal cipher, the final equality is expected to hold with probability 2^{-n} where n is the block length. The probability of the related-key boomerang distinguisher, on the other hand, is approximately p^2q^2 (see [17–20] for details).

Note that the boomerang attack can be generalized to exploit multiple differentials. The success probability then becomes $\hat{p}^2\hat{q}^2$, where $\hat{p}$ and $\hat{q}$ are the square roots of the sums of the squares of the differentials exploited[6].

6.1 Exploiting Nonlinear Differentials

Differentials are often found via linearization, i.e., assuming that integer additions behave as XOR's. One then evaluates the probability of the differential with respect to the probability that each active addition behaves as XOR. This probability equals 2^{-w}, where w is the Hamming weight of the logical OR of the two difference masks, excluding the MSB.

Yet one is not limited to such "linear" differentials, and the best differential—in terms of probability—is not necessarily a linearization, as illustrated by the work of Lipmaa and Moriai [21]: for integer addition, they presented efficient algorithms for computing the probability of any differential, and for finding the optimal differential. The problem was later studied using formal rational series with linear representation [22].

We used the algorithms in [21] to find the differentials of our boomerang attacks. Note that it is not guaranteed that our trails are optimal, for the combination of local optimal differential trails (with respect to their probability) may contribute to a faster increase of the weight than (non-necessarily optimal) linear differentials. Yet our best differentials are not completely linear.

6.2 Related-Key Distinguishers

Like in our previous attacks, we exploit differences in the key and in the plaintext that vanish until the twelfth round (both for the forward and backward

[6]Throughout the paper, our differentials do not make use of this multiple differential approach. One can further improve upon the differentials provided in this work by using this technique.

differentials). Then, we follow a nonlinear differential trail until the middle of the cipher, i.e., between the 16-th and 17-th rounds. Our differential trail for E^{β} has probability $p = 2^{-86}$, and the one for E^{γ} has probability 2^{-113}, leading to a boomerang distinguisher on 34 rounds requiring about $(pq)^{-2} = 2^{398}$ trials (see full version [9]). Note that for the second part, MSB differences are set in the key words k_2 and k_3, and in the tweak words t_0 and t_1 (thus giving no difference in the seventh subkey).

6.3 Known-Related-Key Distinguishers

Although the standard notion of distinguisher requires a secret (key), the notion of *known-key distinguisher* [23] is also relevant to set apart a block cipher from a randomly chosen permutation. Moreover, when a block cipher is used within a compression function, as Threefish is, known-key distinguishers may lead to distinguishers for the hash function because all inputs are known to the adversary. If differences in the keys are used, we shall thus talk of *known-related-key distinguisher*. An example of such distinguisher is the exhibition of input/output pairs that have some specific relation, as presented in [23] for seven rounds of AES-128. Here, we shall consider tuples $(m_1, m_2, m_3, m_4, c_1, c_2, c_3, c_4)$ that satisfy the boomerang property.

To build a known-related-key boomerang distinguisher on Threefish, we consider the decryption function, i.e., we start from the end of the cipher: when the key is known, the attacker can easily find a ciphertext that conforms to the first differential (e.g., to the weight-83 differential at round 35), which we could verify experimentally. In other words, the final differential (including the differences caused by the final key) is "free" when launching the boomerang. When it returns, however, the 2^{83} factor cannot be avoided if we want to exactly follow the differential (which is not strictly necessary to run a distinguisher). We thus obtain a distinguisher on 35-round Threefish-512 with complexity 2^{83} times that of the the related-key distinguisher on 34 rounds, that is, approximately 2^{478} encryptions.

Several tricks may be used to obtain a similar distinguisher at a reduced cost. For example, observing that the first and fourth (resp. second and third) MIX functions of round 34 depend only on the first and second (resp. third and fourth) MIX's of round 35, one can speed-up the search for inputs conforming to the first two rounds of the boomerang.

6.4 Extension to Key-Recovery

We now show how to build a key-recovery attack on top of a boomerang distinguisher for 32-round Threefish-512. We present some preliminary observations before describing and analyzing our attack.

Using notations of Eq. (1): E^{β} starts from the beginning and ends after the key addition in round 16, and E^{γ} starts from round 17 and ends just before the key addition after round 32. Our goal is to recover the *last subkey*. Restricted to 32 rounds, the boomerang distinguisher has probabilities $p = 2^{-86}$ for E^{β} and

$q = 2^{-37}$ for E^{γ}, yielding an overall boomerang probability of $p^2q^2 = 2^{-246}$. We now introduce some notions required to facilitate the analysis of our attack.

Definition 1 (CS-sequence). *Let δ be a 64-bit word of Hamming weight $0 \leq w \leq 64$. The* CS-sequence *of δ is*

$$S_\delta = (|s_0|, |s_1|, \cdots, |s_{w-1}|) ,$$

where $|s_i|$ is the bit length of the i-th block of consecutive zeros in δ finishing with a one.

For example, for $\delta = \texttt{1000010402000000}$ we have

$$\delta = \underbrace{0001}_{s_0}\,\underbrace{0000\;0000\;0000\;0000\;0001}_{s_1}\,\underbrace{0000\;01}_{s_2}\underbrace{00\;0000\;001}_{s_3}0\;0000\;\cdots\;0000 ,$$

and so the CS-sequence of δ is $S_\delta = (|s_0|, |s_1|, |s_2|, |s_3|) = (4, 20, 6, 9)$.

The following result is extensively used in the key recovery attack using boomerang distinguisher, whose proof is provided in the full version of this paper [9].

Theorem 1. *The number of possible differences N_δ after addition of difference δ with zero or $\Delta = \texttt{8000000000000000}$ difference modulo 2^{64} can be directly computed from the CS-sequence of δ as*

$$N_\delta = |s_0| \sum_{(k_1,k_2,\ldots,k_{w-1}) \in \{0,1\}^{w-1}} \prod_{i=1}^{w-1} |s_i|^{k_i} .$$

For instance, if $\delta = \texttt{1000010402000000}$ then

$$\begin{aligned} N_\delta &= 4 \sum_{(k_1,k_2,k_3) \in \{0,1\}^3} (20^{k_1} \times 6^{k_2} \times 9^{k_3}) \\ &= 4 \times (1 + 9 + 6 + (6 \times 9) + 20 + (20 \times 6) + (20 \times 9) + (20 \times 9 \times 6)) \\ &= 4 \times 1470 = 5880 . \end{aligned}$$

Applying Theorem 1, we have the number of possible output differences caused by $\Delta^{\gamma}_{\text{out}}$ just after the key addition followed by the related-key boomerang distinguisher for Threefish-512 is approximately 2^{62}. We obtain this number by multiplying the number of possibilities for each word of the state (see Table 4).

The Attack. Our attack works in three steps: in the first step, we obtain quartets satisfying the related-key boomerang relation; in the second, we recover the partial key by using the possible right quartets obtained from the first step; the last step is the brute force search of the rest of the key. The attack works as follows.

Table 4. Number of possible output differences after the key addition in Threefish-512, for each word. Multiplying these numbers, we obtain in total approximately 2^{62} possible differences.

$v_{32,i}$	$S_{\Delta^{\gamma}_{\text{out}}}$	$N_{\Delta^{\gamma}_{\text{out}}}$
$v_{32,0}$	$(24, 15)$	384
$v_{32,1}$	(32)	32
$v_{32,2}$	(0)	1
$v_{32,3}$	$(4, 20, 6, 9)$	5880
$v_{32,4}$	(1)	1
$v_{32,5}$	$(13, 2, 9, 2, 12, 11, 5)$	957840
$v_{32,6}$	$(13, 11, 30)$	4836
$v_{32,7}$	(14)	14

1. **Find right quartets**
   for $i = 1, \ldots, 2^{248}$
   - Generate a random unique pair of chosen plaintexts (m_1^i, m_2^i) with an $\Delta_{\text{in}}^{\beta}$ difference and encrypt each plaintext with key k^1 and k^2 (having Δ_k^{β} difference) respectively to obtain the corresponding ciphertexts (c_1^i, c_2^i).

   - **for** $j = 1, \ldots, 2^{62}$
     - Set $c_3^{i,j} = c_1^i \oplus \Delta_{\text{out}}^{\prime,j}$ where $\Delta_{\text{out}}^{\prime,j}$ is set to the j-th possible difference caused by $\Delta_{\text{out}}^{\gamma}$.
     - Decrypt $c_3^{i,j}$ with k^3 and obtain the plaintext $m_3^{i,j}$.
     - Store the values $c_3^{i,j}$ and $m_3^{i,j}$.

   - **for** $k = 1, \ldots, 2^{62}$
     - Set $c_4^{i,k} = c_2^i \oplus \Delta_{\text{out}}^{\prime,k}$ where $\Delta_{\text{out}}^{\prime,k}$ is set to the k-th possible difference caused by $\Delta_{\text{out}}^{\gamma}$.
     - Decrypt $c_4^{i,k}$ with k^4 and obtain the plaintext $m_4^{i,k}$.
     - Calculate $M = m_4^{i,k} \oplus \Delta_{\text{in}}^{\beta}$ and check whether M exists among the stored values of $m_3^{i,j}$. If this is the case, store the possible right quartet.

   - Free the memory allocated for the stored values of (possibly wrong) $c_3^{i,j}$ and $m_3^{i,j}$. Increment i.
2. **Recover the partial key**
   For each ciphertext word having a nonzero difference of a (possibly) right quartet (c_1, c_2, c_3, c_4) guess the corresponding output whitening key word $k_{\omega,l}$ for $l = 0, 3, 5, 6$, and check

   $$(c_{1,l} - k_{\omega,l}) \oplus (c_{3,l} - k_{\omega,l}^2) = (c_{2,l} - k_{\omega,l}^3) \oplus (c_{4,l} - k_{\omega,l}^4) = \Delta_{\text{out},l}^{\gamma} ,$$

   where $k_{\omega,l}^2 = k_{\omega,l} \oplus \Delta_{k,l}^{\gamma}$ and $k_{\omega,l}^3 = k_{\omega,l}^4 \oplus \Delta_{k,l}^{\gamma}$. If this is the case, store this $k_{\omega,l}$.
3. **Recover the full key**
   Run an exhaustive search of the remaining bits of the subkey.

Complexity Analysis. The goal of step 1 is to find enough quartets satisfying the related-key boomerang trail. For each distinct 2^{248} plaintext-ciphertext pairs (m_1, m_2) and (c_1, c_2), we correspondingly generate 2^{62} new plaintext-ciphertext pairs (m_3, c_3) and (m_4, c_4) by using the possible number of output differences given in Table 4. We know that a right quartet has to satisfy one of the possible number of output differences Δ'_{out}; hence it is guaranteed to find the right quartet once it exists as we consider all possible combinations. Note that, increasing the number of quartets in that manner does not increase the number of right quartets, the reason simply being the newly generated plaintext-ciphertext pairs (m_3, c_3) and (m_4, c_4) can only have one *root* right plaintext-ciphertext pair (m_1, m_2) and (c_1, c_2). Therefore, the expected number of right quartets is $2^{248} \cdot 2^{-246} = 2^2$. On the other hand, we expect $2^{372} \cdot 2^{-512} = 2^{-140}$ additional false quartets.

The first loop at step 1 requires 2^{62} reduced round Threefish decryptions and approximately $2^{70.5}$ bytes of memory. The second loop can be implemented independently and requires 2^{62} reduced round Threefish decryptions and 2^{62} memory accesses. On the other hand, we need additional memory complexity of $2^{69.5}$ bytes for storing Δ'_{out} values. Therefore, the overall complexity of the first step is bounded by 2^{312} reduced round Threefish decryptions and about 2^{71} bytes of memory. Note that the memory requirement for the surviving quartets is negligible.

Step 2 tries to recover the last subkey by using the quartets that passed the previous step. For each surviving quartet, we guess 64 bits of the final key at each word, decrypt one round and check the output difference $\Delta^{\gamma}_{\text{out},l}$. As the computation at each word can be processed independently, the overall complexity of this step is dominated by the previous step.

The probability that a false combination of quartets and key bits is counted in step 2 is upper bounded by 2^{-2w_l} where w_l is the minimum hamming weight of the corresponding output difference $\Delta'^{,l}_{\text{out}}$. Therefore, the right key is suggested $4 + 2^{-140} \cdot 2^{-2w_l} \approx 4$ times by the right and additional false quartets. On the other hand, a wrong key is expected to be hit $4 \cdot 2^{-2w_l} + 2^{-140} \cdot 2^{-2w_l} \approx 2^{-2}$ times. Note that this only holds for the words having an XOR difference of hamming weight two, for the rest the number of hits is strictly less than 2^{-2}. We can use Poisson distribution to calculate the success rate of our attack. For an expected number of 2^{-2}, the probability that a wrong key is suggested at most once is 0.97. However, the probability that the right key is suggested more than once is more than 0.90. Therefore, we can find the right key or at least eliminate most of the keys with high probability. The complexity of the rest of the attack is dominated by the first step.

7 Conclusion

We applied a wide range of attack strategies to the core algorithm of Skein (the block cipher Threefish-512), culminating with a distinguisher on 35-round Threefish-512, and a key-recovery attack on 32 rounds. Other versions of Threefish are vulnerable to similar attack strategies (for example, our related-key boomerang distinguisher works on up to 33 rounds of Threefish-256). To the best of our knowledge, this is the first application of a key-recovery boomerang attack to an "ARX" algorithm, and also the first application of the boomerang technique to known-key distinguishers.

Despite its relative simplicity, the full Threefish seems to resist state-of-the-art cryptanalytic techniques. Its balanced "ARX" structure combined with large words provides a good balance between diffusion and non-linearity, and avoids any particular structure exploitable by attackers. Using attacks on Threefish to attack the hash function Skein (or its compression function) seems difficult, because of the rather complex mode of operation of Skein. Although none of our attacks directly extends to the hash mode, the pseudorandomness of Threefish is required to validate the security proofs on Skein. Hence, 36 or more rounds of Threefish seem to be required to provide optimal security.

Future works might apply the recent rebound attack [24] to Threefish, although it looks difficult to combine it with the trick discussed in §§3.1; this forces the attacker to use specific differences. Another research direction relates to optimization of boomerang known- or chosen-key distinguishers.

References

1. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of LNCS., Springer (2005) 19–35
2. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In Shoup, V., ed.: CRYPTO. Volume 3621 of LNCS., Springer (2005) 17–36
3. Cannière, C.D., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In Lai, X., Chen, K., eds.: ASIACRYPT. Volume 4284 of LNCS., Springer (2006) 1–20
4. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Naor, M., ed.: EUROCRYPT. Volume 4515 of LNCS., Springer (2007) 1–22
5. NIST: FIPS 180-2 Secure Hash Standard (2002)
6. NIST: Cryptographic Hash Competition http://www.nist.gov/hash-competition.
7. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Submission to NIST (2008)
8. Bellare, M., Kohno, T., Lucks, S., Ferguson, N., Schneier, B., Whiting, D., Callas, J., Walker, J.: Provable Security Support for the Skein Hash Family. http://www.skein-hash.info/sites/default/files/skein-proofs.pdf Draft, February 18, 2009.
9. Aumasson, J.P., Çağdaş Çalık, Meier, W., Özen, O., Phan, R.C.W., cı K.V.: Improved Cryptanalysis of Skein. Cryptology ePrint Archive (2009)
10. Biham, E., Chen, R.: Near-Collisions of SHA-0. In Franklin, M.K., ed.: CRYPTO. Volume 3152 of LNCS., Springer (2004) 290–305
11. NIST: SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2001)
12. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In Knudsen, L.R., ed.: FSE. Volume 1636 of LNCS., Springer (1999) 124–138

13. Knudsen, L.R.: DEAL - a 128-bit Block Cipher. Technical Report 151, University of Bergen (1998) Submitted as an AES candidate.
14. Jakimoski, G., Desmedt, Y.: Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In Matsui, M., Zuccherato, R.J., eds.: Selected Areas in Cryptography. Volume 3006 of LNCS., Springer (2003) 208–221
15. Biham, E., Dunkelman, O., Keller, N.: Related-Key Impossible Differential Attacks on 8-Round AES-192. In Pointcheval, D., ed.: CT-RSA. Volume 3860 of LNCS., Springer (2006) 21–33
16. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In Nyberg, K., ed.: FSE. Volume 5086 of LNCS., Springer (2008) 470–488
17. Wagner, D.: The Boomerang Attack. In Knudsen, L.R., ed.: FSE. Volume 1636 of LNCS., Springer (1999) 156–170
18. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of LNCS., Springer (2005) 507–525
19. Biham, E., Dunkelman, O., Keller, N.: New Combined Attacks on Block Ciphers. In Gilbert, H., Handschuh, H., eds.: FSE. Volume 3557 of LNCS., Springer (2005) 126–144
20. Dunkelman, O.: Techniques for Cryptanalysis of Block Ciphers. PhD thesis, Technion, Israel (February 2006)
21. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In Matsui, M., ed.: FSE. Volume 2355 of LNCS., Springer (2001) 336–350
22. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive-Or. In Roy, B.K., Meier, W., eds.: FSE. Volume 3017 of LNCS., Springer (2004) 317–331
23. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In Kurosawa, K., ed.: ASIACRYPT. Volume 4833 of LNCS., Springer (2007) 315–324
24. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Dunkelman, O., ed.: FSE. Volume 5665 of Lecture Notes in Computer Science., Springer (2009) 260–276

A Conforming Pair for the 4-Round Differential

When the key and the tweak are zero, the following two message blocks conform to the differential described in §§3.2:

```
E979D16280002004  32B29AE900000000  D921590E00000000  5771CC9000000400
A62FF22800000000  484B245000040080  D3BEA4E800008010  7A72784300000000

A971917200100020  72B2DAE980002004  DD61588E01000400  5331CC1000000000
A62FF22800040090  C84B245000000000  D1BEA4E800000000  FA72784300008010
```

B Examples of Near Collisions

We provide an example of near collision on 459 bits for the reduced compression function of Skein's UBI mode. Both inputs always have $k_0 = \cdots = k_4 = k_7 = 0$, and

$$k_5 = \texttt{C0DEC0DEC0DEC0DE}.$$

On the 16-round compression function, the first input has message block

```
E979D16280002004  32B29AE900000000  D921590E00000000  5771CC9000000400
A62FF22800000000  484B245000040080  D3BEA4E800008010  7A72784300000000
```

and

$k_6 =$ `6B9B2C1000000000` $t_0 =$ `3F213F213F213F22` $t_1 =$ `9464D3F000000000`

The second input has message block

```
A971917200100020  72B2DAE980002004  DD61588E01000400  5331CC1000000000
A62FF22800040090  C84B245000000000  D1BEA4E800000000  FA72784300008010
```

and

$k_6 =$ `6B9B2C1000000000` $t_0 =$ `BF213F213F213F22` $t_1 =$ `9464D3F000000000`

The corresponding digests are respectively

```
2A6DE91E3E8CDE3B  BADAF451F59D3145  7C298A43FB73463F  D8309C9E9E2594D5
35431D226A2022E3  0EA42EB45F9EEEB9  DF038EECD6504300  588A798B1266D67A
```

and

```
6A65A80EBE9CFF1F  FADAB450759D1141  78618AC3FA73463F  5C709C1A9E2590D5
B5431D226A242273  8EAE2FF45B9A6A39  5D038EECD650C310  D08E788B1266576A
```