J. Cryptol. (2011) 24: 1–23
DOI: 10.1007/s00145-010-9058-x

Journal of
CRYPTOLOGY

Practical Collisions for EnRUPT*

Sebastiaan Indesteege† and Bart Preneel

Department of Electrical Engineering ESAT/COSIC, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
sebastiaan.indesteege@esat.kuleuven.be
and
Interdisciplinary Institute for BroadBand Technology (IBBT), Ghent, Belgium

Communicated by Antoine Joux

Received 27 September 2009 and revised 28 December 2009
Online publication 5 February 2010

Abstract. The EnRUPT hash functions were proposed by O'Neil, Nohl and Henzen as candidates for the SHA-3 competition, organised by NIST. The proposal contains seven concrete hash functions, each with a different digest length. We present a practical collision attack on each of these seven EnRUPT variants. The time complexity of our attack varies from 2^{36} to 2^{40} round computations, depending on the EnRUPT variant, and the memory requirements are negligible. We demonstrate that our attack is practical by giving an actual collision example for EnRUPT-256.

Key words. EnRUPT, SHA-3 candidate, Hash function, Collision attack.

1. Introduction

Cryptographic hash functions are important cryptographic primitives that are employed in a vast number of applications, such as digital signatures and commitment schemes. They are expected to possess several security properties, one of which is *collision resistance*. Informally, collision resistance means that it should be hard to find two distinct messages $m \neq m'$ that hash to the same value, i.e. $h(m) = h(m')$.

Many popular hash functions, such as MD5, SHA-1 and SHA-2 share some common design principles. The recent advances in the cryptanalysis of these hash functions have raised serious concerns regarding their long-term security. This motivates the design of new hash functions, based on different design strategies. The National Institute of Standards and Technology (NIST) has decided to hold a public competition, the SHA-3 competition, to develop a new cryptographic hash function standard [6].

* This paper is an extended version of http://dx.doi.org/10.1007/978-3-642-03317-9_15. This paper was solicited by the Editor-in-Chief as one of the best papers from Fast Sofware Encryption 2009, based on the recommendation of the program committee.

† F.W.O. Research Assistant, Fund for Scientific Research—Flanders (Belgium).

© International Association for Cryptologic Research 2010

The EnRUPT hash functions were proposed by O'Neil, Nohl and Henzen [7] as candidates in this SHA-3 competition. The proposal contains seven concrete EnRUPT variants, each with a different digest length. Khovratovich et al. [4] presented a theoretical preimage attack on EnRUPT, with a time complexity of 2^{480} and requiring about 2^{384} memory elements.

In this paper, we analyse EnRUPT and show that none of the proposed EnRUPT variants is collision resistant. We present a practical collision attack requiring only 2^{36} to 2^{40} EnRUPT round computations, depending on the EnRUPT variant. This is significantly less than the approximately $2^{n/2}$ hash computations required for a generic collision attack on an n-bit hash function, based on the birthday paradox.

The structure of this paper is as follows. A short description of EnRUPT is given in Sect. 2. Section 3 introduces the basic strategy used to find collisions for EnRUPT, which is based on the work on SHA by Chabaud and Joux [2] and Rijmen and Oswald [11]. Sections 4, 5 and 6 apply this basic attack strategy to EnRUPT, step by step. Our results, including an example collision for EnRUPT-256, are presented in Sect. 7. Finally, Sect. 8 concludes.

2. Description of EnRUPT

In this section, we give a short description of the seven EnRUPT variants that were proposed as SHA-3 candidates [7]. All share the same structure and use the same round function. The only differences lie in the parameters used. Table 1 gives the values of these parameters for each EnRUPT variant.

2.1. *The EnRUPT Hash Functions*

The structure shared by all EnRUPT hash functions can be split into four phases: preprocessing, message processing, finalisation and output. Figure 1 contains a description of the EnRUPT hash functions in pseudocode.

In the preprocessing phase (lines 2–4) the input message is padded to be a multiple of w bits, where w is the word size. Depending on the EnRUPT variant, the word size w is 32 or 64 bits, see Table 1. The padded message is then split into an integer number of w-bit words m_i.

Table 1. EnRUPT Parameters.

EnRUPT variant	Digest length h	Word size w	Parallelisation level P	Security parameter s	Number of state words H
EnRUPT-128	128 bits	32 bits	2	4	8
EnRUPT-160	160 bits	32 bits	2	4	10
EnRUPT-192	192 bits	32 bits	2	4	12
EnRUPT-224	224 bits	64 bits	2	4	8
EnRUPT-256	256 bits	64 bits	2	4	8
EnRUPT-384	384 bits	64 bits	2	4	12
EnRUPT-512	512 bits	64 bits	2	4	16

```
1:  function EnRUPT (M)
2:      /* Preprocessing */
3:      m_0, ..., m_t ← M||1||0^{w−(|M|+1 mod w)}   s.t.   ∀i, 0 ≤ i ≤ t : |m_i| = w
4:      d_0, ..., d_{P−1}, x_0, ..., x_{H−1}, r ← 0, ..., 0
5:      /* Message processing */
6:      for i = 0 to n do
7:          ⟨d, x, r⟩ ← round(⟨d, x, r⟩, m_i)
8:      end for
9:      /* Finalisation */
10:     ⟨d, x, r⟩ ← round(⟨d, x, r⟩, uint_w(|M|))
11:     for i = 1 to H do
12:         ⟨d, x, r⟩ ← round(⟨d, x, r⟩, 0)
13:     end for
14:     /* Output */
15:     for i = 0 to h/w − 1 do
16:         ⟨d, x, r⟩ ← round(⟨d, x, r⟩, 0)
17:         o_i ← d_0
18:     end for
19:     return o_0||···||o_{h/w−1}
20: end function
```

Fig. 1. The EnRUPT hash function.

The internal state of EnRUPT consists of several w-bit words: H state words x_i, P 'delta accumulators' d_i, and a round counter r. All of these are initialised to zero. The parameter P is equal to 2 for all seven EnRUPT variants. The value of H depends on the digest length, as indicated in Table 1.

Then, in the message processing phase (lines 5–8), the round function is called once for each w-bit padded message word m_i. Each call to the round function updates the internal state $\langle d, x, r\rangle$. A detailed description of the EnRUPT round function is given in the next section, Sect. 2.2.

After all message words have been processed, a finalisation is performed (lines 9–13). The EnRUPT round function is called once with the length of the (unpadded) message, represented as a w-bit unsigned integer.[1] Then, H blank rounds, i.e. calls to the round function with a zero message word input, are performed.

Finally, in the output phase (lines 14–18), the message digest is generated one w-bit word at a time. The EnRUPT round function is called h/w times and, after each call, the content of the 'delta accumulator' d_0 is output.

2.2. *The EnRUPT Round Function*

The EnRUPT round function is based entirely on a number of simple operations on words of w bits, such as bit shifts, bit rotations, exclusive OR and addition modulo 2^w. Figure 2 gives a description of the EnRUPT round function in pseudocode. The round function consists of $s \cdot P$ identical steps, where s and P are parameters of the hash function. As indicated in Table 1, $s = 4$ and $P = 2$ for all seven proposed EnRUPT variants. Thus, the EnRUPT round function consists of eight steps.

[1] Note that the EnRUPT specification [7] only states that the message length should be included, not how this is to be done exactly. The EnRUPT reference implementation uses one w-bit word for the message length, which implies that the EnRUPT variants for which $w = 32$ can only handle up to $2^{32} - 1$ bits in this implementation. Note that the results presented in this paper are independent of the details of the padding.

```
 1: function round (⟨d, x, r⟩, m)
 2:     for i = 0 to s · P − 1 do                    /* An iteration of this loop is a "step" */
 3:         /* Compute indices */
 4:         α ← r + (i + 1 mod P) mod H
 5:         β ← r + i + 2P mod H
 6:         γ ← r + i + P mod H
 7:         ξ ← r + i mod H
 8:         /* Compute intermediate f */
 9:         e ← ((x_α ≪ 1) ⊕ x_β ⊕ d_{i mod P} ⊕ uint_w(r + i)) ⋙ w/4
10:         f ← (e ≪ 3) ⊞ e                          /* Multiplication with 9 modulo 2^w */
11:         /* Update state */
12:         x_γ ← x_γ ⊕ f
13:         d_{i mod P} ← d_{i mod P} ⊕ x_ξ ⊕ f
14:     end for
15:     r ← r + s · P
16:     d_{P−1} ← d_{P−1} ⊕ m                        /* Message word injection */
17:     return ⟨d, x, r⟩
18: end function
```

Fig. 2. The EnRUPT round function.

In each step, several words of the state are selected (lines 4–7) and combined into an intermediate value f (lines 9–10). Note that line 10 could equally be described as a multiplication with 9 modulo 2^w. The intermediate value f is then used to update one state word, x_γ, and one 'delta accumulator', $d_{i \bmod P}$ (lines 12–13).

After all steps have been performed, the round counter is incremented by the number of steps that were carried out, i.e. $s \cdot P$ (line 15). Finally, the input message word m is injected into one word of the internal state, the 'delta accumulator' d_{P-1} (line 16).

3. Basic Attack Strategy

This section gives an overview of the linearization method for finding collision differential characteristics for a hash function, which we use to attack EnRUPT in this work. This method was introduced by Chabaud and Joux [2], who applied it to SHA-0 and simplified variants thereof. Later, it was extended further and applied to SHA-1 by Rijmen and Oswald [11].

A Linear Hash Function Consider a hypothetical hash function that consists only of linear operations over GF(2). When the input messages are restricted to a certain length, each output bit can be written as an affine function of the input bits. The *difference* in each output bit is given by a linear function of the differences in the input bits, as the constants (if any) cancel out. A message difference that leads to a collision can be found by equating the output differences to zero, and solving the resulting system of linear equations over GF(2), for instance using Gauss elimination. Any pair of messages with this difference will result in a collision.

Linearising a Nonlinear Hash Function Actual cryptographic hash functions contain (also) nonlinear components, so this method no longer applies. However, we may still be able to *approximate* the nonlinear components by linear ones and construct a linear approximation of the entire hash function. For our purpose, a good linear approximation

$\lambda(x)$ of a nonlinear function $\gamma(x)$ is such that its differential behaviour is close to that of $\gamma(x)$. More formally, the equation

$$\gamma(x \oplus \Delta) \oplus \gamma(x) = \lambda(x \oplus \Delta) \oplus \lambda(x) = \lambda(\Delta) \tag{1}$$

should hold for a relatively large fraction of values x. For instance, an addition modulo 2^w could be approximated by a simple XOR operation, i.e. ignoring the carries.

Finding Collisions A *differential characteristic* consists of a message difference and a list of the differences in all (relevant) intermediate values. For the linear approximation, it is easy to find a differential characteristic that leads to a collision with probability one. But for the actual hash function, this probability will be (much) lower.

If the differential behaviour of all the nonlinear components corresponds to that of the linear approximations they were replaced with, i.e. if (1) holds simultaneously for each nonlinear component, we say that the differential characteristic is followed. In this case, the message pair under consideration will not only collide for the linearised hash function, but also for the original, nonlinear hash function. Such a message pair is called a *conforming* message pair.

Hence, a procedure for finding a collision for the nonlinear hash function could be to find a differential characteristic leading to collisions for a linearised variant of the hash function. Then, a message pair conforming to the differential characteristic is searched. In order to lower the complexity of the attack, it is important to maximise the probability that the differential characteristic is followed, i.e. we need to find a *good* differential characteristic.

4. Linearising EnRUPT

We now apply this general strategy to EnRUPT. Recall the description of the EnRUPT round function in Fig. 2. Note that only the modular addition in line 10 is not linear over GF(2). Indeed, the computation of the indices in lines 4–7 and the update of the round counter in line 15 do not depend on the message being hashed and can thus be precomputed. The same holds for the inclusion of the round counter in line 9, which can be seen as an XOR with a constant. The other operations are all linear over GF(2).

Replacing the modular addition in line 10 with an XOR operation yields a linearised round function, which we refer to as the EnRUPTl round function. The EnRUPTl hash function, i.e. the hash function built on this linearised round function, also consists solely of GF(2)-linear components.

5. The Collision Search

During the collision search phase, many collisions for EnRUPTl are constructed, and a collision for EnRUPT is searched among them. Since only the modular additions (line 10 of Fig. 2) were approximated by XOR, these are the only places where the propagation of differences could differ between EnRUPT-$\mathcal{L}$ and EnRUPT. Instead of checking for a collision at the output, we can immediately check if the difference at the output of each modular addition, i.e. the difference Δf in the intermediate value f, still matches the differential characteristic.

5.1. *An Observation on EnRUPT*

We now make an important observation on the structure of the EnRUPT hash function. It is possible to find a conforming message pair for a given differential characteristic one round at a time.

Consider the message word m_i, which is injected into the 'delta accumulator' d_{P-1} at the end of round i. In the first $(P-1)$ steps of the next round, d_{P-1} is not used, so m_i can not influence the behaviour of the modular additions in these steps. Starting from the Pth step of round $(i+1)$, however, m_i does have an influence.

We can search for a value for m_i such that the differential characteristic is followed up to and including the first $(P-1)$ steps of round $(i+2)$. Starting with the Pth step of round $(i+2)$, the next message word, m_{i+1} also influences the modular additions. Thus, we can keep m_i fixed, and use the new freedom available in m_{i+1} to ensure that the differential characteristic is also followed for the next $s \cdot P$ steps.

This drastically reduces the expected number of trials required to find a collision. Let p_i denote the probability that the differential characteristic is followed in a block of $s \cdot P$ consecutive steps, starting at the Pth step of a round. Because we can construct a conforming message pair one word at a time, the expected number of trials is $\sum_i 1/p_i$ rather than $\prod_i 1/p_i$. In other words, the complexities associated with each block of $s \cdot P$ steps should be added together, rather than multiplied. This possibility was ignored in the security analysis of EnRUPT [7], leading to the incorrect conclusion that attacks based on linearization do not apply.

5.2. *Accelerating the Collision Search*

A simple optimisation can be made to the collision search, which allows us to ignore the probability associated with one step in each round. This optimisation is analogous to Wang's 'single message modification', which was first introduced in the context of MD5 and other hash functions of the MD4-family [13].

Consider the Pth step of a round. In this step, the 'delta accumulator' d_{P-1}, to which a new message word m was XORed at the end of the previous round, is used for the first time. More precisely, it is used in line 9 of Fig. 2 to compute the intermediate value e. Note, however, that these computations can be inverted. We can choose the value of e, and compute backwards to find what the message word m should be to arrive at this value of e:

$$\begin{aligned} m &= d_{P-1} \oplus d_{P-1}^{\text{prev}} \\ &= (e \lll w/4) \oplus (x_\alpha \ll 1) \oplus x_\beta \oplus \text{uint}_w(r+P-1) \oplus d_{P-1}^{\text{prev}}. \end{aligned} \tag{2}$$

Here, d_{P-1}^{prev} is the (known) value of d_{P-1} in the previous round, just before the message word m was added to it.

The values of e which ensure that the difference propagation of the modular addition in line 10 of Fig. 2 corresponds to that of its linear approximation can be efficiently enumerated as follows. Consider a binary tree representing all possible values for e. Each layer of the tree determines one more bit of e, starting from the least significant bit. This tree is walked in a depth-first fashion, backtracking as soon as the difference propagation is not as desired. Indeed, the difference propagation in the lower bits does

not depend on the more significant bits, so this backtracking strategy effectively skips over all bad values for e.

Thus, rather than randomly picking values for m, we can efficiently sample *good* values for e in this step, and compute backwards to find the corresponding m. This ensures that the first modular addition affected by a message word m will always exhibit the desired propagation of differences. Thus, the Pth step of every round can be ignored in the estimation of the complexity of the attack.

6. Finding Good Differential Characteristics

The key to lowering the attack complexity is to find a good differential characteristic, i.e. a characteristic which is likely to be followed for the nonlinear hash function. A generic approach to this problem, based on finding low weight codewords in a linear code, was proposed by Rijmen and Oswald [11] and extended by Pramstaller et al. [10]. In this section, we show how to apply this approach to EnRUPT.

6.1. *Coding Theory*

As observed by Rijmen and Oswald [11], all of the differential characteristics leading to a collision for the linearised hash function can be seen as the codewords of a linear code.

Consider the EnRUPT-$\mathcal{L}$ hash function with a h-bit output length, and the message input restricted to messages of t message words. Since it is affine over GF(2), it is possible to express the difference in the output as a linear function of the difference in the input message m:

$$[\Delta o]_{1\times h} = [\Delta m]_{1\times tw} \cdot [\mathbf{O}]_{tw\times h}. \tag{3}$$

As the modular additions, or rather the multiplications with 9, in the EnRUPT round function are approximated, we are also interested in the differences that enter each of these operations. For EnRUPT restricted to t message blocks, there are $t \cdot s \cdot P$ such operations in total. Hence, we can combine the input differences to these operations in a $1 \times tsPw$ bit vector Δe. Again, for the linear approximation, Δe is simply a linear function of the message difference Δm:

$$[\Delta e]_{1\times tsPw} = [\Delta m]_{1\times tw} \cdot [\mathbf{E}]_{tw\times tsPw}. \tag{4}$$

Putting this together results in a linear code described by the following generator matrix

$$\mathbf{G} = [\mathbf{I}_{tw\times tw} | \mathbf{E}_{tw\times tsPw} | \mathbf{O}_{tw\times h}]. \tag{5}$$

Each codeword contains a message difference, the input differences to all approximated modular additions, and finally the output difference.

Thus, each codeword is in fact a differential characteristic for EnRUPT-$\mathcal{L}$, and all differential characteristics for EnRUPT-$\mathcal{L}$ are codewords of this code. To restrict ourselves to collision differentials, i.e. differential characteristics ending in a zero output difference, we can use Gauss elimination to force the h rightmost columns of the generator matrix G to zero.

It is well known that the differential behaviour of modular addition can be well approximated by that of XOR when the Hamming weight of the input difference, ignoring the most significant bit, is small [2,5,10,11]. As the input differences to the modular additions are part of the codewords, we will attempt to find a codeword with a low Hamming weight in this part of the codeword.

6.2. *Low Weight Codewords*

To find low weight codewords, we used a simple and straightforward algorithm that is based on the assumption that a codeword of very low weight exists in the code. For our purposes, this is a reasonable assumption, as only a very low weight codeword will lead to an attack faster than a generic attack. The algorithm is related to the algorithm of Canteaut and Chabaud [1] and the algorithm used to find low weight codewords for linearised SHA-1 by Pramstaller et al. [10].

Let G be the generator matrix of the linear code as in (5). We randomly select a set I of (appropriate) columns of the generator matrix G and force them to zero using Gauss elimination, until only d rows remain, where d is a parameter of the algorithm. Then, the remaining space of 2^d codewords is searched exhaustively. This procedure is repeated until a codeword of sufficiently low weight is encountered. By replacing only the 'oldest' column(s) in I, instead of restarting from the beginning every time, the algorithm can be implemented efficiently in practice.

If a codeword of very low weight exists in the code, it is likely that all of the columns in the randomly constructed set I will coincide with zeroes in the codeword, which implies that the codeword will be found in the exhaustive search phase. In the case of the codes originating from the seven linearised EnRUPT variants we consider, this algorithm finds a codeword of very low weight in a matter of minutes on a PC. Repeated runs of the algorithm always find the same codewords, so it is reasonable to assume that these are indeed the best codewords we can find.

6.3. *Estimating the Attack Complexity*

Actually, the weight of a codeword is only a heuristic for the attack complexity resulting from the corresponding differential. Codewords with a lower weight are expected to result in a lower attack complexity, but we can easily enhance our algorithm to optimise the actual attack complexity, rather than just a crude heuristic.

The Differential Probability The probability that a differential characteristic is followed, is determined by the differences that are input to each of the multiplications with 9 (line 10 in Fig. 2), which were approximated using XOR operations. Denote by $\mathrm{DP}^{\times 9}(\Delta)$ the probability that the propagation of differences through this nonlinear operation coincides with that of its linear approximation:

$$\mathrm{DP}^{\times 9}(\Delta) = \Pr_x\big[(x \times 9) \oplus ((x \oplus \Delta) \times 9) = \Delta \oplus (\Delta \ll 3)\big]. \tag{6}$$

The differential probability of modular addition was studied by Lipmaa and Moriai [5]. Applying their results to this situation, and taking into account that the three least sig-

nificant bits of $(x \ll 3)$ are always zero, we find the following estimate for $\mathrm{DP}^{\times 9}(\Delta)$:

$$\mathrm{DP}^{\times 9}(\Delta) \approx 2^{-\mathrm{wt}((\Delta \vee (\Delta \ll 3)) \wedge \mathtt{0111\cdots111000}_b)}. \tag{7}$$

Even though this estimate ignores the dependency between x and $(x \ll 3)$, this confirms the intuition that a difference Δ with a low Hamming weight (ignoring the most significant bit and the three least significant bits) results in a large probability $\mathrm{DP}^{\times 9}(\Delta)$. We used this as a heuristic to find a good differential characteristic: we want to minimise the Hamming weight of the relevant parts of the differences that are input to the modular additions. In other words, we want to find a low weight codeword of the aforementioned linear code, where only the bits that impact $\mathrm{DP}^{\times 9}(\Delta)$ are counted.

Exact Computation of the Differential Probability Computing the exact value of $\mathrm{DP}^{\times 9}(\Delta)$ for any given difference Δ can be done by counting all the values x for which the difference propagation is as predicted by the linear approximation. We now show how this can be done in an efficient way. While this is very useful for evaluating the precise attack complexity, it lacks the clear intuition we can gather from (7).

For w-bit words, the definition of $\mathrm{DP}^{\times 9}(\Delta)$ given in (6) can be restated as

$$\mathrm{DP}^{\times 9}(\Delta) = \frac{\#\{x \in \{0,1\}^w \mid (x \times 9) \oplus ((x \oplus \Delta) \times 9) = \Delta \oplus (\Delta \ll 3)\}}{2^w}. \tag{8}$$

Now, consider how the computation of $y = (x \times 9) = x \oplus (x \ll 3)$ is performed at the bit level. Let x_i denote the ith bit of x, where x_0 is the least significant bit. Then the following equations can be derived:

$$\begin{cases} y_i = x_i \oplus x_{i-3} \oplus c_i, \\ c_{i+1} = \mathrm{maj}(x_i, x_{i-3}, c_i). \end{cases} \tag{9}$$

Here, the bits c_i represent the carry bits in the modular addition. By definition, the first carry bit c_0 is zero. The majority function $\mathrm{maj}(\cdot)$ is defined by

$$\mathrm{maj}(a, b, c) = ab \oplus bc \oplus ac. \tag{10}$$

Let Δx_i be the XOR difference in the ith bit of x, and similar for other differences. Then, we find

$$\begin{cases} \Delta y_i = \Delta x_i \oplus \Delta x_{i-3} \oplus \Delta c_i, \\ \Delta c_{i+1} = \mathrm{maj}(x_i, x_{i-3}, c_i) \oplus \mathrm{maj}(x_i \oplus \Delta x_i, x_{i-3} \oplus \Delta x_{i-3}, c_i \oplus \Delta c_i). \end{cases} \tag{11}$$

Since the output difference of the multiplication is approximated by $\Delta x \oplus (\Delta x \ll 3)$, it follows from the first equation of (11) that we require $\Delta c_i = 0$ for $0 \le i < w$. Note that only the knowledge of x_i, x_{i-3} and c_i is required to evaluate (11) when the input difference Δx is fixed, and the carry bits have no difference.

Hence, this can be represented efficiently in a trellis where the computations relating to one bit slice are represented by one segment in the trellis. Each node in the trellis represents, for a certain bit position, the values of the input carry c_i and the values of

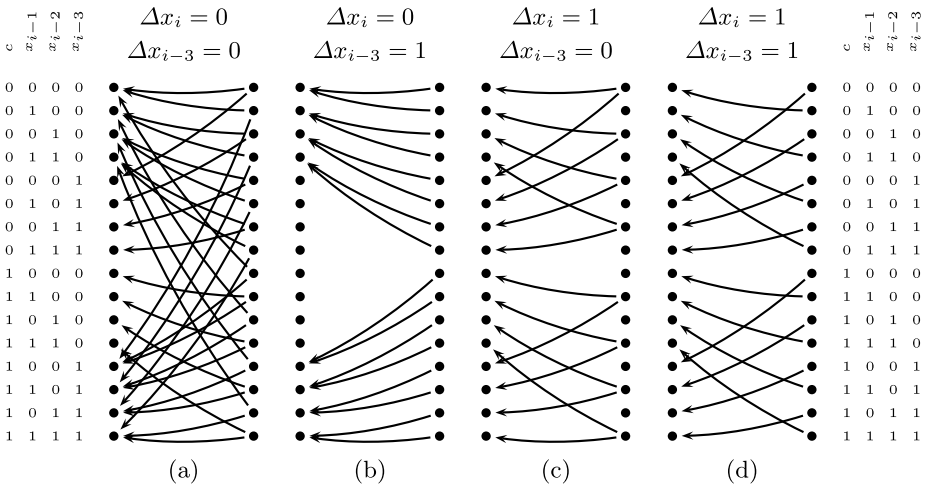

Fig. 3. Trellis segments used in the calculation of $\mathrm{DP}^{\times 9}$.

the three most recent bits, x_{i-1}, x_{i-2} and x_{i-3}. Since the differences in x are fixed a priori, and we do not allow differences in the carry, it is possible to compute the arcs in the segment as for each value of the bit x_i, the next node can be computed. Indeed, the next node is identified by c_{i+1}, which can be computed from (9) and x_i, x_{i-1} and x_{i-2}. Note, however, that, except for the most significant bit, we require $\Delta c_{i+1} = 0$ for our approximation to hold. This can be checked using (11), and only arcs satisfying this condition are kept. The full trellis is the concatenation of segments out of the four possibilities shown in Fig. 3. The input difference Δx, which is fixed, determines which segments are used. Note that the trellis segment for the most significant bit contains all arcs regardless of Δx, as the condition prohibiting an output carry differences is no longer required there.

For each value of x for which no carry differences occur, there is a path in the trellis starting at the node $\langle 0, 0, 0, 0\rangle$ at the input of least significant bit slice, which we refer to as the source node, and ending at one of the nodes at the output of the most significant bit slice, the goal nodes. Hence, we can evaluate (8) by simply counting the number of such paths. This can be done using an algorithm that bears similarity to the Viterbi algorithm [12] and is an example of dynamic programming. Let $f(N)$ denote the number of paths from the source node to a node N. It is straightforward to see that this is equal to

$$f(N) = \sum_{\mathrm{arc}(X,N)} f(X), \tag{12}$$

where $\mathrm{arc}(X, N)$ denotes that there is an arc from node X to node N. For the source node S, we initialise $f(S) = 1$. Then the recurrence (12) can be used to evaluate the number of paths to all nodes in the trellis. Finally, the sum of the number of paths to each of the goal nodes allows to compute $\mathrm{DP}^{\times 9}$ using (8).

Computing the Attack Complexity Let $p_{r,i}$ be the differential probability associated with the modular addition in step i of round r of the differential characteristic. Recall the observation made in Sect. 5.1, i.e. finding a conforming message pair can be done one round at a time, or rather one message word at a time, as this does not coincide precisely with the round boundaries. Taking this into account, the complexity of finding the jth word of a conforming message pair can thus be computed as

$$C_j = \left(\prod_{i=P-1}^{sP-1} \frac{1}{p_{j+1,i}} \right) \left(\prod_{i=0}^{P-2} \frac{1}{p_{j+2,i}} \right). \tag{13}$$

Due to the acceleration technique presented in Sect. 5.2, we are guaranteed that the differential behaviour of the modular addition in step $P-1$ of each round will be as desired. Thus, we can set $p_{r,P-1} = 1$. With the default EnRUPT parameters ($P = 2$ and $s = 4$, see Table 1), this then becomes

$$C_j = \frac{1}{p_{j+1,2}} \cdot \frac{1}{p_{j+1,3}} \cdot \frac{1}{p_{j+1,4}} \cdot \frac{1}{p_{j+1,5}} \cdot \frac{1}{p_{j+1,6}} \cdot \frac{1}{p_{j+1,7}} \cdot \frac{1}{p_{j+2,0}}. \tag{14}$$

Finally, as was explained in Sect. 5.1, note that each message word can be found independently of the previous ones, due to the newly available degrees of freedom in each message word. Hence, the overall attack complexity can simply be computed as the sum of these round complexities:

$$C_{\text{tot}} = \sum_{j=0}^{t} C_j. \tag{15}$$

Note that, given a differential characteristic, it is easy to compute the associated attack complexity. Hence, when searching for a good differential characteristic using the algorithm described in Sect. 6.2, we can use the actual attack complexity instead of the weight of the codeword. The algorithm still implicitly uses the weight of a codeword as a heuristic, but now attempts to optimise the actual attack complexity directly.

7. Results and Discussion

We constructed differential characteristics for each of the seven EnRUPT variants in the EnRUPT SHA-3 proposal [7]. Table 2 lists the attack complexity and the length of the best characteristic we found for each variant. For the sake of clarity, the key parameters of each EnRUPT variant are repeated from Table 1. Recall that we fixed the length of the characteristic a priori. Note, however, that nothing prevents our search algorithm from proposing a shorter characteristic, padded with rounds without any difference, which we also observed in practice. We experimented with (much) longer maximum characteristic lengths, but found no better long characteristics.

The time complexities vary from 2^{36} to 2^{40} round computations, depending on the EnRUPT variant, which is remarkable. It means that the collision resistance in absolute terms of each of these EnRUPT variants is more or less the same, regardless of the digest length. Relative to the expected collision resistance of approximately $2^{n/2}$ for an n-bit

Table 2. Summary of our attacks. Only the best attack is listed for each EnRUPT variant.

EnRUPT variant	Word size w [bits]	State size H [words]	Estimated time complexity [EnRUPT rounds]	Length of differential [message words]
EnRUPT-128	32	8	$2^{36.04}$	6
EnRUPT-160	32	10	$2^{37.78}$	7
EnRUPT-192	32	12	$2^{38.33}$	8
EnRUPT-224	64	8	$2^{37.02}$	6
EnRUPT-256	64	8	$2^{37.02}$	6
EnRUPT-384	64	12	$2^{39.63}$	8
EnRUPT-512	64	16	$2^{38.46}$	10

hash function, however, the (relative) collision resistance of EnRUPT is much worse for the variants with a longer digest length than for those with a shorter digest length.

Tables 3, 4, 5, 6, 7, 8 list our differential characteristics for each of the seven EnRUPT variants. Note that the same characteristic applies to EnRUPT-224 and EnRUPT-256 (Table 6) as both functions share the same parameter settings, see Table 1.

The format of these tables is as follows. Each line in the table corresponds to one step of the EnRUPT round function. The difference in the input (Δe) and the output (Δf) of the multiplication with 9 in that step is indicated. Also, the message word differences are shown at the end of each round. Note that the word size is 32 bits for some EnRUPT variants, and 64 bits for others. The table also includes the differential probabilities of each step, which were used to compute the attack complexity. A star ('$\star$') indicates that the differential probability can be ignored in that step because of the technique presented in Sect. 5.2. The product of the step probabilities is given for eight consecutive steps. Note that these do not coincide with the rounds, as was discussed in Sect. 6.3.

A collision example for EnRUPT-256, obtained using the characteristic from Table 6, is given in Table 9. This example was computed on a cluster of AMD Opteron 250 processors running at 2.4 GHz. The total computational effort was 237 CPU-days. This is roughly 1000 times what one would expect if one were to count just the time spent doing EnRUPT rounds using an optimised implementation of EnRUPT. This discrepancy is explained by a lack of optimisation of the implementation of the attack algorithm, a simple but somewhat wasteful approach to parallelisation, and a considerable amount of redundant work due to a bug in the initial attack implementation. However, such practical issues are to be expected in a first implementation, and we opted to balance the programming effort and CPU time, rather than pursuing the fastest possible implementation.

Discussion In response to these collision attacks, the designers of EnRUPT proposed to double the s parameter to 8, or to increase it even further to be equal to the H-parameter, see Table 1 [8,9]. As a consequence of this, the number of steps between two message word injections is at least doubled. Experiments with these EnRUPT variants indicate that this tweak seems to be effective at stopping the attacks described in this paper. For EnRUPT-256 with $s = 6$, we were still able to find a differential with an associated attack complexity of about 2^{110} EnRUPT rounds, which is still below the birthday bound. For higher values of the s parameter, all the differential characteristics

Table 3. Differential characteristic for EnRUPT-128.

Round	Step	$\Delta e \rightarrow \Delta f$	$\mathrm{DP}^{\times 9}$	Totals
Inject message word difference $\Delta m_{-1} = \mathtt{00000800}_x$				
0	0	$\mathtt{00000000}_x \rightarrow \mathtt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$\mathtt{00000008}_x \rightarrow \mathtt{00000048}_x$	$\star$	
	2	$\mathtt{90000000}_x \rightarrow \mathtt{10000000}_x$	$2^{-0.85}$	
	3	$\mathtt{48000008}_x \rightarrow \mathtt{08000048}_x$	$2^{-3.85}$	
	4	$\mathtt{90000000}_x \rightarrow \mathtt{10000000}_x$	$2^{-0.85}$	
	5	$\mathtt{48280008}_x \rightarrow \mathtt{09680048}_x$	$2^{-6.92}$	
	6	$\mathtt{9002d000}_x \rightarrow \mathtt{10145000}_x$	$2^{-6.43}$	
	7	$\mathtt{00296808}_x \rightarrow \mathtt{01622848}_x$	$2^{-10.39}$	
Inject message word difference $\Delta m_0 = \mathtt{00228000}_x$				
1	0	$\mathtt{9002d000}_x \rightarrow \mathtt{10145000}_x$	$2^{-6.43}$	$\mathbf{2^{-35.72}}$
	1	$\mathtt{00296800}_x \rightarrow \mathtt{01622800}_x$	$\star$	
	2	$\mathtt{9002d000}_x \rightarrow \mathtt{10145000}_x$	$2^{-6.43}$	
	3	$\mathtt{48280000}_x \rightarrow \mathtt{09680000}_x$	$2^{-4.92}$	
	4	$\mathtt{9002d000}_x \rightarrow \mathtt{10145000}_x$	$2^{-6.43}$	
	5	$\mathtt{00080000}_x \rightarrow \mathtt{00480000}_x$	$2^{-1.85}$	
	6	$\mathtt{90024000}_x \rightarrow \mathtt{10104000}_x$	$2^{-3.69}$	
	7	$\mathtt{48092000}_x \rightarrow \mathtt{08402000}_x$	$2^{-5.71}$	
Inject message word difference $\Delta m_1 = \mathtt{00228800}_x$				
2	0	$\mathtt{90024000}_x \rightarrow \mathtt{10104000}_x$	$2^{-3.69}$	$\mathbf{2^{-32.73}}$
	1	$\mathtt{00084800}_x \rightarrow \mathtt{004a0800}_x$	$\star$	
	2	$\mathtt{90024000}_x \rightarrow \mathtt{10104000}_x$	$2^{-3.69}$	
	3	$\mathtt{48096800}_x \rightarrow \mathtt{08422800}_x$	$2^{-8.45}$	
	4	$\mathtt{90024000}_x \rightarrow \mathtt{10104000}_x$	$2^{-3.69}$	
	5	$\mathtt{00200000}_x \rightarrow \mathtt{01200000}_x$	$2^{-1.85}$	
	6	$\mathtt{90000000}_x \rightarrow \mathtt{10000000}_x$	$2^{-0.85}$	
	7	$\mathtt{48200000}_x \rightarrow \mathtt{09200000}_x$	$2^{-3.26}$	
Inject message word difference $\Delta m_2 = \mathtt{00020800}_x$				
3	0	$\mathtt{90000000}_x \rightarrow \mathtt{10000000}_x$	$2^{-0.85}$	$\mathbf{2^{-22.65}}$
	1	$\mathtt{00292000}_x \rightarrow \mathtt{01602000}_x$	$\star$	
	2	$\mathtt{90009000}_x \rightarrow \mathtt{10041000}_x$	$2^{-3.70}$	
	3	$\mathtt{48296800}_x \rightarrow \mathtt{09622800}_x$	$2^{-9.68}$	
	4	$\mathtt{90009000}_x \rightarrow \mathtt{10041000}_x$	$2^{-3.70}$	
	5	$\mathtt{00084800}_x \rightarrow \mathtt{004a0800}_x$	$2^{-4.59}$	
	6	$\mathtt{90009000}_x \rightarrow \mathtt{10041000}_x$	$2^{-3.70}$	
	7	$\mathtt{48080000}_x \rightarrow \mathtt{08480000}_x$	$2^{-3.71}$	
Inject message word difference $\Delta m_3 = \mathtt{00020000}_x$				
4	0	$\mathtt{90009000}_x \rightarrow \mathtt{10041000}_x$	$2^{-3.70}$	$\mathbf{2^{-32.76}}$
	1	$\mathtt{00080008}_x \rightarrow \mathtt{00480048}_x$	$\star$	
	2	$\mathtt{00000000}_x \rightarrow \mathtt{00000000}_x$	$2^{-0.00}$	
	3	$\mathtt{00080008}_x \rightarrow \mathtt{00480048}_x$	$2^{-3.85}$	
	4	$\mathtt{00000000}_x \rightarrow \mathtt{00000000}_x$	$2^{-0.00}$	
	5	$\mathtt{48084808}_x \rightarrow \mathtt{084a0848}_x$	$2^{-8.47}$	
	6	$\mathtt{00000000}_x \rightarrow \mathtt{00000000}_x$	$2^{-0.00}$	
	7	$\mathtt{48084808}_x \rightarrow \mathtt{084a0848}_x$	$2^{-8.47}$	

Table 3. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	$DP^{\times 9}$	Totals
Inject message word difference $\Delta m_4 = \texttt{00020000}_x$				
5	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-20.79}}$
	1	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$\star$	
	$\vdots$	$\vdots \rightarrow \vdots$	$\vdots$	
	7	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 4. Differential characteristic for EnRUPT-160.

Round	Step	$\Delta e \rightarrow \Delta f$	$DP^{\times 9}$	Totals
Inject message word difference $\Delta m_{-1} = \texttt{00000400}_x$				
0	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$\texttt{00000004}_x \rightarrow \texttt{00000024}_x$	$\star$	
	2	$\texttt{48000000}_x \rightarrow \texttt{08000000}_x$	$2^{-1.85}$	
	3	$\texttt{24000004}_x \rightarrow \texttt{04000024}_x$	$2^{-2.85}$	
	4	$\texttt{48000000}_x \rightarrow \texttt{08000000}_x$	$2^{-1.85}$	
	5	$\texttt{24140004}_x \rightarrow \texttt{04b40024}_x$	$2^{-5.92}$	
	6	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	
	7	$\texttt{2414b404}_x \rightarrow \texttt{04b11424}_x$	$2^{-10.70}$	
Inject message word difference $\Delta m_0 = \texttt{00114000}_x$				
1	0	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	$\mathbf{2^{-38.04}}$
	1	$\texttt{0014b400}_x \rightarrow \texttt{00b11400}_x$	$\star$	
	2	$\texttt{00016800}_x \rightarrow \texttt{000a2800}_x$	$2^{-5.59}$	
	3	$\texttt{2414b400}_x \rightarrow \texttt{04b11400}_x$	$2^{-9.68}$	
	4	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	
	5	$\texttt{24000000}_x \rightarrow \texttt{04000000}_x$	$2^{-1.85}$	
	6	$\texttt{48000000}_x \rightarrow \texttt{08000000}_x$	$2^{-1.85}$	
	7	$\texttt{0000b400}_x \rightarrow \texttt{00051400}_x$	$2^{-5.59}$	
Inject message word difference $\Delta m_1 = \texttt{00114400}_x$				
2	0	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	$\mathbf{2^{-39.41}}$
	1	$\texttt{0004b400}_x \rightarrow \texttt{00211400}_x$	$\star$	
	2	$\texttt{48012000}_x \rightarrow \texttt{08082000}_x$	$2^{-4.69}$	
	3	$\texttt{00042400}_x \rightarrow \texttt{00250400}_x$	$2^{-4.59}$	
	4	$\texttt{00012000}_x \rightarrow \texttt{00082000}_x$	$2^{-2.85}$	
	5	$\texttt{24042400}_x \rightarrow \texttt{04250400}_x$	$2^{-6.45}$	
	6	$\texttt{48012000}_x \rightarrow \texttt{08082000}_x$	$2^{-4.69}$	
	7	$\texttt{24042400}_x \rightarrow \texttt{04250400}_x$	$2^{-6.45}$	
Inject message word difference $\Delta m_2 = \texttt{00000000}_x$				
3	0	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	$\mathbf{2^{-34.42}}$
	1	$\texttt{0010b400}_x \rightarrow \texttt{00951400}_x$	$\star$	
	2	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	
	3	$\texttt{00100000}_x \rightarrow \texttt{00900000}_x$	$2^{-1.85}$	
	4	$\texttt{48000000}_x \rightarrow \texttt{08000000}_x$	$2^{-1.85}$	
	5	$\texttt{0014b400}_x \rightarrow \texttt{00b11400}_x$	$2^{-8.36}$	

Table 4. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	6	$\texttt{00004800}_x \rightarrow \texttt{00020800}_x$	$2^{-2.85}$	
	7	$\texttt{2414b400}_x \rightarrow \texttt{04b11400}_x$	$2^{-9.68}$	
Inject message word difference $\Delta m_3 = \texttt{00010400}_x$				
4	0	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	$\mathbf{2^{-33.99}}$
	1	$\texttt{24002400}_x \rightarrow \texttt{04010400}_x$	$\star$	
	2	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	
	3	$\texttt{00002400}_x \rightarrow \texttt{00010400}_x$	$2^{-2.85}$	
	4	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	
	5	$\texttt{00042400}_x \rightarrow \texttt{00250400}_x$	$2^{-4.59}$	
	6	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	
	7	$\texttt{00042400}_x \rightarrow \texttt{00250400}_x$	$2^{-4.59}$	
Inject message word difference $\Delta m_4 = \texttt{00010000}_x$				
5	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-26.12}}$
	1	$\texttt{24042404}_x \rightarrow \texttt{04250424}_x$	$\star$	
	2	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	
	3	$\texttt{00040004}_x \rightarrow \texttt{00240024}_x$	$2^{-2.85}$	
	4	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	
	5	$\texttt{24042404}_x \rightarrow \texttt{04250424}_x$	$2^{-7.47}$	
	6	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	
	7	$\texttt{24042404}_x \rightarrow \texttt{04250424}_x$	$2^{-7.47}$	
Inject message word difference $\Delta m_5 = \texttt{00010000}_x$				
6	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-17.79}}$
	1	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$\star$	
	$\vdots$	$\vdots \quad \rightarrow \quad \vdots$	$\vdots$	
	7	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 5. Differential characteristic for EnRUPT-192.

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
Inject message word difference $\Delta m_{-1} = \texttt{00000800}_x$				
0	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$\texttt{00000008}_x \rightarrow \texttt{00000048}_x$	$\star$	
	2	$\texttt{90000000}_x \rightarrow \texttt{10000000}_x$	$2^{-0.85}$	
	3	$\texttt{48000008}_x \rightarrow \texttt{08000048}_x$	$2^{-3.85}$	
	4	$\texttt{90000000}_x \rightarrow \texttt{10000000}_x$	$2^{-0.85}$	
	5	$\texttt{48280008}_x \rightarrow \texttt{09680048}_x$	$2^{-6.92}$	
	6	$\texttt{9002d000}_x \rightarrow \texttt{10145000}_x$	$2^{-6.43}$	
	7	$\texttt{48296808}_x \rightarrow \texttt{09622848}_x$	$2^{-11.70}$	
Inject message word difference $\Delta m_0 = \texttt{00228000}_x$				
1	0	$\texttt{9002d000}_x \rightarrow \texttt{10145000}_x$	$2^{-6.43}$	$\mathbf{2^{-37.03}}$
	1	$\texttt{48296800}_x \rightarrow \texttt{09622800}_x$	$\star$	
	2	$\texttt{0002d000}_x \rightarrow \texttt{00145000}_x$	$2^{-5.58}$	
	3	$\texttt{48296800}_x \rightarrow \texttt{09622800}_x$	$2^{-9.68}$	

Table 5. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	4	$\texttt{0002d000}_x \rightarrow \texttt{00145000}_x$	$2^{-5.58}$	
	5	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	
	6	$\texttt{90000000}_x \rightarrow \texttt{10000000}_x$	$2^{-0.85}$	
	7	$\texttt{48016800}_x \rightarrow \texttt{080a2800}_x$	$2^{-7.43}$	
	Inject message word difference $\Delta m_1 = \texttt{00228800}_x$			
2	0	$\texttt{90000000}_x \rightarrow \texttt{10000000}_x$	$2^{-0.85}$	$\mathbf{2^{-37.41}}$
	1	$\texttt{00016800}_x \rightarrow \texttt{000a2800}_x$	⋆	
	2	$\texttt{9002d000}_x \rightarrow \texttt{10145000}_x$	$2^{-6.43}$	
	3	$\texttt{00016800}_x \rightarrow \texttt{000a2800}_x$	$2^{-5.59}$	
	4	$\texttt{9002d000}_x \rightarrow \texttt{10145000}_x$	$2^{-6.43}$	
	5	$\texttt{48080000}_x \rightarrow \texttt{08480000}_x$	$2^{-3.71}$	
	6	$\texttt{00024000}_x \rightarrow \texttt{00104000}_x$	$2^{-2.85}$	
	7	$\texttt{48084800}_x \rightarrow \texttt{084a0800}_x$	$2^{-6.45}$	
	Inject message word difference $\Delta m_2 = \texttt{00000000}_x$			
3	0	$\texttt{00024000}_x \rightarrow \texttt{00104000}_x$	$2^{-2.85}$	$\mathbf{2^{-34.30}}$
	1	$\texttt{48092000}_x \rightarrow \texttt{08402000}_x$	⋆	
	2	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	3	$\texttt{48092000}_x \rightarrow \texttt{08402000}_x$	$2^{-5.71}$	
	4	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	5	$\texttt{00212000}_x \rightarrow \texttt{01282000}_x$	$2^{-4.58}$	
	6	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	7	$\texttt{00200000}_x \rightarrow \texttt{01200000}_x$	$2^{-1.85}$	
	Inject message word difference $\Delta m_3 = \texttt{00000000}_x$			
4	0	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	$\mathbf{2^{-26.93}}$
	1	$\texttt{48296800}_x \rightarrow \texttt{09622800}_x$	⋆	
	2	$\texttt{00009000}_x \rightarrow \texttt{00041000}_x$	$2^{-2.85}$	
	3	$\texttt{48292000}_x \rightarrow \texttt{09602000}_x$	$2^{-6.92}$	
	4	$\texttt{00009000}_x \rightarrow \texttt{00041000}_x$	$2^{-2.85}$	
	5	$\texttt{48000000}_x \rightarrow \texttt{08000000}_x$	$2^{-1.85}$	
	6	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	7	$\texttt{48004800}_x \rightarrow \texttt{08020800}_x$	$2^{-4.70}$	
	Inject message word difference $\Delta m_4 = \texttt{00020800}_x$			
5	0	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	$\mathbf{2^{-26.56}}$
	1	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	⋆	
	2	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	3	$\texttt{00004800}_x \rightarrow \texttt{00020800}_x$	$2^{-2.85}$	
	4	$\texttt{90009000}_x \rightarrow \texttt{10041000}_x$	$2^{-3.70}$	
	5	$\texttt{48080000}_x \rightarrow \texttt{08480000}_x$	$2^{-3.71}$	
	6	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	
	7	$\texttt{48080000}_x \rightarrow \texttt{08480000}_x$	$2^{-3.71}$	
	Inject message word difference $\Delta m_5 = \texttt{00020000}_x$			
6	0	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	$\mathbf{2^{-17.66}}$
	1	$\texttt{48084808}_x \rightarrow \texttt{084a0848}_x$	⋆	
	2	$\texttt{00000000}_x \rightarrow \texttt{00000000}_x$	$2^{-0.00}$	

Table 5. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	3	$00080008_x \rightarrow 00480048_x$	$2^{-3.85}$	
	4	$00000000_x \rightarrow 00000000_x$	$2^{-0.00}$	
	5	$48084808_x \rightarrow 084a0848_x$	$2^{-8.47}$	
	6	$00000000_x \rightarrow 00000000_x$	$2^{-0.00}$	
	7	$48084808_x \rightarrow 084a0848_x$	$2^{-8.47}$	
Inject message word difference $\Delta m_6 = 00020000_x$				
7	0	$00000000_x \rightarrow 00000000_x$	$2^{-0.00}$	$\mathbf{2^{-20.79}}$
	1	$00000000_x \rightarrow 00000000_x$	$\star$	
	$\vdots$	$\vdots \rightarrow \vdots$	$\vdots$	
	7	$00000000_x \rightarrow 00000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 6. Differential characteristic for EnRUPT-224 or EnRUPT-256.

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
Inject message word difference $\Delta m_{-1} = 0000000008000000_x$				
0	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$0000000000000800_x \rightarrow 0000000000004800_x$	$\star$	
	2	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	3	$4800000000000800_x \rightarrow 0800000000004800_x$	$2^{-3.70}$	
	4	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	5	$4800280000000800_x \rightarrow 0801680000004800_x$	$2^{-7.28}$	
	6	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	7	$0000280168000800_x \rightarrow 0001680a28004800_x$	$2^{-11.02}$	
Inject message word difference $\Delta m_0 = 0000002280000000_x$				
1	0	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	$\mathbf{2^{-36.56}}$
	1	$0000280168000000_x \rightarrow 0001680a28000000_x$	$\star$	
	2	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	3	$4800280000000000_x \rightarrow 0801680000000000_x$	$2^{-5.43}$	
	4	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	5	$0000080000000000_x \rightarrow 0000480000000000_x$	$2^{-1.85}$	
	6	$9000000240000000_x \rightarrow 1000001040000000_x$	$2^{-3.70}$	
	7	$4800080120000000_x \rightarrow 0800480820000000_x$	$2^{-6.54}$	
Inject message word difference $\Delta m_1 = 0000002288000000_x$				
2	0	$9000000240000000_x \rightarrow 1000001040000000_x$	$2^{-3.70}$	$\mathbf{2^{-34.08}}$
	1	$0000080048000000_x \rightarrow 0000480208000000_x$	$\star$	
	2	$9000000240000000_x \rightarrow 1000001040000000_x$	$2^{-3.70}$	
	3	$4800080168000000_x \rightarrow 0800480a28000000_x$	$2^{-9.28}$	
	4	$9000000240000000_x \rightarrow 1000001040000000_x$	$2^{-3.70}$	
	5	$0000200000000000_x \rightarrow 0001200000000000_x$	$2^{-1.85}$	
	6	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	7	$4800200000000000_x \rightarrow 0801200000000000_x$	$2^{-3.70}$	

Table 6. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	Inject message word difference $\Delta m_2 = 0000000208000000_x$			
3	0	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	$\mathbf{2^{-23.91}}$
	1	$0000280120000000_x \rightarrow 0001680820000000_x$	$\star$	
	2	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	3	$4800280168000000_x \rightarrow 0801680a28000000_x$	$2^{-11.02}$	
	4	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	5	$0000080048000000_x \rightarrow 0000480208000000_x$	$2^{-4.70}$	
	6	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	7	$4800080000000000_x \rightarrow 0800480000000000_x$	$2^{-3.70}$	
	Inject message word difference $\Delta m_3 = 0000000200000000_x$			
4	0	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	$\mathbf{2^{-34.19}}$
	1	$0000080000000800_x \rightarrow 0000480000004800_x$	$\star$	
	2	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	3	$0000080000000800_x \rightarrow 0000480000004800_x$	$2^{-3.70}$	
	4	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	5	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
	Inject message word difference $\Delta m_4 = 0000000200000000_x$			
5	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-20.49}}$
	1	$0000000000000000_x \rightarrow 0000000000000000_x$	$\star$	
	$\vdots$	$\vdots \rightarrow \vdots$	$\vdots$	
	7	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 7. Differential characteristic for EnRUPT-384.

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	Inject message word difference $\Delta m_{-1} = 0000000008000000_x$			
0	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$0000000000000800_x \rightarrow 0000000000004800_x$	$\star$	
	2	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	3	$4800000000000800_x \rightarrow 0800000000004800_x$	$2^{-3.70}$	
	4	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	5	$4800280000000800_x \rightarrow 0801680000004800_x$	$2^{-7.28}$	
	6	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	7	$4800280168000800_x \rightarrow 0801680a28004800_x$	$2^{-12.87}$	
	Inject message word difference $\Delta m_0 = 0000002280000000_x$			
1	0	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	$\mathbf{2^{-38.41}}$
	1	$4800280168000000_x \rightarrow 0801680a28000000_x$	$\star$	
	2	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	
	3	$4800280168000000_x \rightarrow 0801680a28000000_x$	$2^{-11.02}$	
	4	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	
	5	$4800000168000000_x \rightarrow 0800000a28000000_x$	$2^{-7.43}$	

Table 7. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	6	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	7	$4800000168000000_x \rightarrow 0800000a28000000_x$	$2^{-7.43}$	
	Inject message word difference $\Delta m_1 = 0000002288000000_x$			
2	0	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	$\mathbf{2^{-38.75}}$
	1	$0000000168000000_x \rightarrow 0000000a28000000_x$	$\star$	
	2	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	3	$0000000168000000_x \rightarrow 0000000a28000000_x$	$2^{-5.58}$	
	4	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	5	$4800080000000000_x \rightarrow 0800480000000000_x$	$2^{-3.70}$	
	6	$0000000240000000_x \rightarrow 0000001040000000_x$	$2^{-2.85}$	
	7	$4800080048000000_x \rightarrow 0800480208000000_x$	$2^{-6.54}$	
	Inject message word difference $\Delta m_2 = 0000000000000000_x$			
3	0	$0000000240000000_x \rightarrow 0000001040000000_x$	$2^{-2.85}$	$\mathbf{2^{-34.39}}$
	1	$4800080120000000_x \rightarrow 0800480820000000_x$	$\star$	
	2	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	3	$4800080120000000_x \rightarrow 0800480820000000_x$	$2^{-6.54}$	
	4	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	5	$0000200120000000_x \rightarrow 0001200820000000_x$	$2^{-4.70}$	
	6	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	7	$0000200000000000_x \rightarrow 0001200000000000_x$	$2^{-1.85}$	
	Inject message word difference $\Delta m_3 = 0000000000000000_x$			
4	0	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	$\mathbf{2^{-27.87}}$
	1	$4800280168000000_x \rightarrow 0801680a28000000_x$	$\star$	
	2	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	
	3	$4800280120000000_x \rightarrow 0801680820000000_x$	$2^{-8.28}$	
	4	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	
	5	$4800000000000000_x \rightarrow 0800000000000000_x$	$2^{-1.85}$	
	6	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	7	$4800000048000000_x \rightarrow 0800000208000000_x$	$2^{-4.70}$	
	Inject message word difference $\Delta m_4 = 0000000208000000_x$			
5	0	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	$\mathbf{2^{-27.91}}$
	1	$0000000000000000_x \rightarrow 0000000000000000_x$	$\star$	
	2	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	3	$0000000048000000_x \rightarrow 0000000208000000_x$	$2^{-2.85}$	
	4	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	5	$4800080000000000_x \rightarrow 0800480000000000_x$	$2^{-3.70}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800080000000000_x \rightarrow 0800480000000000_x$	$2^{-3.70}$	
	Inject message word difference $\Delta m_5 = 0000000200000000_x$			
6	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-17.63}}$
	1	$4800080048000800_x \rightarrow 0800480208004800_x$	$\star$	
	2	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	3	$0000080000000800_x \rightarrow 0000480000004800_x$	$2^{-3.70}$	
	4	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	

Table 7. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	5	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
	Inject message word difference $\Delta m_6 = 0000000200000000_x$			
7	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-20.49}}$
	1	$0000000000000000_x \rightarrow 0000000000000000_x$	$\star$	
	$\vdots$	$\vdots \rightarrow \vdots$	$\vdots$	
	7	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 8. Differential characteristic for EnRUPT-512 (part 1).

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	Inject message word difference $\Delta m_{-1} = 0000000008000000_x$			
0	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$
	1	$0000000000000800_x \rightarrow 0000000000004800_x$	$\star$	
	2	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	3	$4800000000000800_x \rightarrow 0800000000004800_x$	$2^{-3.70}$	
	4	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	5	$4800280000000800_x \rightarrow 0801680000004800_x$	$2^{-7.28}$	
	6	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	7	$4800280168000800_x \rightarrow 0801680a28004800_x$	$2^{-12.87}$	
	Inject message word difference $\Delta m_0 = 0000002280000000_x$			
1	0	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	$\mathbf{2^{-38.41}}$
	1	$4800280168000000_x \rightarrow 0801680a28000000_x$	$\star$	
	2	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	
	3	$0000280168000000_x \rightarrow 0001680a28000000_x$	$2^{-9.17}$	
	4	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	
	5	$0000000168000000_x \rightarrow 0000000a28000000_x$	$2^{-5.58}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800000000000000_x \rightarrow 0800000000000000_x$	$2^{-1.85}$	
	Inject message word difference $\Delta m_1 = 0000002288000000_x$			
2	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-27.77}}$
	1	$4800000000000000_x \rightarrow 0800000000000000_x$	$\star$	
	2	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	3	$4800000168000000_x \rightarrow 0800000a28000000_x$	$2^{-7.43}$	
	4	$9000000000000000_x \rightarrow 1000000000000000_x$	$2^{-0.85}$	
	5	$0000000168000000_x \rightarrow 0000000a28000000_x$	$2^{-5.58}$	
	6	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	
	7	$0000000168000000_x \rightarrow 0000000a28000000_x$	$2^{-5.58}$	
	Inject message word difference $\Delta m_2 = 0000000000000000_x$			
3	0	$90000002d0000000_x \rightarrow 1000001450000000_x$	$2^{-6.43}$	$\mathbf{2^{-33.16}}$
	1	$4800000000000000_x \rightarrow 0800000000000000_x$	$\star$	
	2	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	

Table 8. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	3	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	4	$00000002d0000000_x \rightarrow 0000001450000000_x$	$2^{-5.58}$	
	5	$0000080168000000_x \rightarrow 0000480a28000000_x$	$2^{-7.43}$	
	6	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	
	7	$4800080048000000_x \rightarrow 0800480208000000_x$	$2^{-6.54}$	
	Inject message word difference $\Delta m_3 = 0000000000000000_x$			
4	0	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	$\mathbf{2^{-30.84}}$
	1	$4800080048000000_x \rightarrow 0800480208000000_x$	$\star$	
	2	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	3	$4800080120000000_x \rightarrow 0800480820000000_x$	$2^{-6.54}$	
	4	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	5	$0000200120000000_x \rightarrow 0001200820000000_x$	$2^{-4.70}$	
	6	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	7	$0000200048000000_x \rightarrow 0001200208000000_x$	$2^{-4.70}$	
	Inject message word difference $\Delta m_4 = 0000000000000000_x$			
5	0	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	$\mathbf{2^{-30.72}}$
	1	$4800280120000000_x \rightarrow 0801680820000000_x$	$\star$	
	2	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	3	$0000280120000000_x \rightarrow 0001680820000000_x$	$2^{-6.43}$	
	4	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	5	$0000000048000000_x \rightarrow 0000000208000000_x$	$2^{-2.85}$	
	6	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	
	7	$4800000048000000_x \rightarrow 0800000208000000_x$	$2^{-4.70}$	
	Inject message word difference $\Delta m_5 = 0000000000000000_x$			
6	0	$0000000090000000_x \rightarrow 0000000410000000_x$	$2^{-2.85}$	$\mathbf{2^{-19.67}}$
	1	$4800000000000000_x \rightarrow 0800000000000000_x$	$\star$	
	2	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	3	$4800000048000000_x \rightarrow 0800000208000000_x$	$2^{-4.70}$	
	4	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	5	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	6	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	
	7	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	Inject message word difference $\Delta m_6 = 0000000208000000_x$			
7	0	$9000000090000000_x \rightarrow 1000000410000000_x$	$2^{-3.70}$	$\mathbf{2^{-19.48}}$
	1	$4800000048000000_x \rightarrow 0800000208000000_x$	$\star$	
	2	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	3	$0000000048000000_x \rightarrow 0000000208000000_x$	$2^{-2.85}$	
	4	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	5	$0000080048000000_x \rightarrow 0000480208000000_x$	$2^{-4.70}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800080000000000_x \rightarrow 0800480000000000_x$	$2^{-3.70}$	
	Inject message word difference $\Delta m_7 = 0000000200000000_x$			
8	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-11.24}}$
	1	$4800080048000800_x \rightarrow 0800480208004800_x$	$\star$	
	2	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	

Table 8. (*Continued.*)

Round	Step	$\Delta e \rightarrow \Delta f$	DP$^{\times 9}$	Totals
	3	$0000080000000800_x \rightarrow 0000480000004800_x$	$2^{-3.70}$	
	4	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	5	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
	6	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	
	7	$4800080048000800_x \rightarrow 0800480208004800_x$	$2^{-8.39}$	
Inject message word difference $\Delta m_8 = 0000000200000000_x$				
9	0	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-20.49}}$
	1	$0000000000000000_x \rightarrow 0000000000000000_x$	$\star$	
	$\vdots$	$\vdots \rightarrow \vdots$	$\vdots$	
	7	$0000000000000000_x \rightarrow 0000000000000000_x$	$2^{-0.00}$	$\mathbf{2^{-0.00}}$

Table 9. A collision example for EnRUPT-256.

M	13_x	$c8_x$	$4b_x$	45_x	62_x	70_x	17_x	$6e_x$
	04_x	$f9_x$	31_x	$7e_x$	$c3_x$	$6c_x$	$e7_x$	$d3_x$
	$e1_x$	21_x	78_x	$6a_x$	34_x	74_x	11_x	19_x
	$7f_x$	64_x	$a3_x$	$c9_x$	40_x	07_x	75_x	76_x
	$a1_x$	$4f_x$	90_x	86_x	fd_x	$c7_x$	33_x	$4a_x$
	41_x	$3a_x$	76_x	91_x	96_x	06_x	$2c_x$	$a1_x$
M'	13_x	$c8_x$	$4b_x$	45_x	$6a_x$	70_x	17_x	$6e_x$
	04_x	$f9_x$	31_x	$5c_x$	43_x	$6c_x$	$e7_x$	$d3_x$
	$e1_x$	21_x	78_x	48_x	bc_x	74_x	11_x	19_x
	$7f_x$	64_x	$a3_x$	cb_x	48_x	07_x	75_x	76_x
	$a1_x$	$4f_x$	90_x	84_x	fd_x	$c7_x$	33_x	$4a_x$
	41_x	$3a_x$	76_x	93_x	96_x	06_x	$2c_x$	$a1_x$
EnRUPT-256(M) =	bd_x	67_x	51_x	$7c_x$	$a6_x$	$c0_x$	41_x	20_x
EnRUPT-256(M') =	82_x	$e0_x$	$3b_x$	74_x	$5f_x$	fc_x	$4a_x$	64_x
	$e9_x$	$f0_x$	92_x	$c2_x$	58_x	$c3_x$	98_x	$b8_x$
	44_x	$9a_x$	fe_x	cb_x	$7f_x$	$c8_x$	$6f_x$	72_x

we could find would result in attack complexities that are far beyond than the birthday bound, and thus should not be considered to be real attacks.

Note that the failure of this heuristic attack method for $s = 8$ or $s = H$ does not preclude the possibility of attacks based on linearization. Our experiments only show that it is unlikely that the particular attack method used in this work can be applied directly to EnRUPT with $s \geq 8$.

8. Conclusion

We presented collision attacks on all seven variants of the EnRUPT hash function [7] that were proposed as candidates to the NIST SHA-3 competition [6]. The attacks require negligible memory and have time complexities ranging from 2^{36} to 2^{40} EnRUPT round computations, depending on the EnRUPT variant. The practicality of the attacks has been demonstrated with an example collision for EnRUPT-256.

Acknowledgements

This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The collision example for EnRUPT-256 was obtained utilising high performance computational resources provided by the University of Leuven, http://ludit.kuleuven.be/hpc.

References

[1] A. Canteaut, F. Chabaud, A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Inf. Theory* **44**(1), 367–378 (1998)

[2] F. Chabaud, A. Joux, Differential collisions in SHA-0, in *Advances in Cryptology—CRYPTO 1998*. Lecture Notes in Computer Science, vol. 1462 (Springer, Berlin, 1998), pp. 56–71

[3] S. Indesteege, B. Preneel, Practical collisions for EnRUPT, in *Fast Software Encryption—FSE 2009*. Lecture Notes in Computer Science, vol. 5665 (Springer, Berlin, 2009), pp. 246–259

[4] D. Khovratovich, I. Nikolic, R.-P. Weinmann, Meet-in-the-middle attacks on SHA-3 candidates, in *Fast Software Encryption—FSE 2009*. Lecture Notes in Computer Science, vol. 5665 (Springer, Berlin, 2009), pp. 228–245

[5] H. Lipmaa, S. Moriai, Efficient algorithms for computing differential properties of addition, in *Fast Software Encryption—FSE 2001*. Lecture Notes in Computer Science, vol. 2355 (Springer, Berlin, 2002), pp. 336–350

[6] National Institute, of Standards and Technology, *Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family*. Federal Register, vol. 72, no. 212, pp. 62212–62220, November 2007

[7] S. O'Neil, K. Nohl, L. Henzen, EnRUPT hash function specification. Submission to the NIST SHA-3 competition, 2008. Available online at http://www.enrupt.com/SHA3/

[8] S. O'Neil, Personal communication, 20 January, 2009

[9] S. O'Neil, EnRUPT. The First SHA-3 Candidate Conference, Leuven, Belgium, February 25–29, 2009. Available online at http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/documents/EnRUPT_2009.pdf

[10] N. Pramstaller, C. Rechberger, V. Rijmen, Exploiting coding theory for collision attacks on SHA-1, in *Cryptography and Coding, 10th IMA International Conference*. Lecture Notes in Computer Science, vol. 3796 (Springer, Berlin, 2005), pp. 78–95

[11] V. Rijmen, E. Oswald, Update on SHA-1, in *Topics in Cryptology—CT-RSA 2005*. Lecture Notes in Computer Science, vol. 3376 (Springer, Berlin, 2005), pp. 58–71

[12] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **13**(3), 260–269 (1967)

[13] X. Wang, H. Yu, How to break MD5 and other hash functions, in *Advances in Cryptology—EUROCRYPT 2005*. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 19–35