Cryptanalysis of Edon-R

Dmitry Khovratovich, Ivica Nikolić, and Ralf-Philipp Weinmann

University of Luxembourg

Abstract. We present various types of attacks on the hash family Edon-R. In a free start attack scenario, with the initial chaining value not fixed, all three main attacks (collisions, second preimage, and preimage) can be launched on Edon-R with negligible effort. In these attacks we exploit the asymmetrical diffusion of the chaining values in the compression function. Also, by partially inverting the compression function and fixing one part of the chaining value, we launch a meet-in-the-middle attack on Edon-R-n to find real preimages. The attack requires $2^{\frac{2n}{3}}$ effort and the same amount of memory. The attacks are applicable to all digest sizes.

1 Description of Edon-R

The hash family Edon-R uses the well known Merkle-Damgard design principle. The intermediate hash value is rather large, two times the digest length[1]. Further, we will describe only Edon-256. The chaining value H_i is composed of two block of 256 bits each, i.e. $H_i = (H_i^1, H_i^2)$. The message input M_i, for the compression function, is also composed of two blocks, i.e. $M_i = (M_i^1, M_i^2)$. Let $Edon_{256}$ be the compression function. Then the new chaining value is produced as follows:

$$H_{i+1} = (H_{i+1}^1, H_{i+1}^2) = Edon_{256}(M_i^1, M_i^2, H_i^1, H_i^2)$$

The hash value of a message is the value of second block of the last chaining value.

Internally, the state of $Edon_{256}$ has two blocks, A and B, of 256 bits each. $Edon_{256}$ makes eight updates, by applying the quasigroup operation $Q(x, y)$, to one of these blocks. The exact definition of the quasigroup operation can be found in [1]. With A_i and B_i we will denote the values of these blocks after the i-th update in the compression function. Hence, each input pair (H_i, M_i) generates internal state blocks $(A_1, B_1), (A_2, B_2), \ldots, (A_8, B_8)$. The new chaining value (the output of the compression function) H_{i+1} is the value of the blocks (A_8, B_8).

2 Simple observations

We present a few simple observations that are further used in the attacks.

[1] Edon-224 and Edon-384 has 512 and 1024 bits chaining values, respectively.

Observation 1. The quasigroup operation $Q(x, y)$ of Edon-R can be presented as:

$$Q(x, y) = \mu(x) + \nu(y),$$

where the functions $\mu(x)$ and $\nu(y)$ can easily be inverted. The exact definitions of these functions are irrelevant for the attack.

In order to invert any of these functions, we need to solve two systems of linear equations. Hence we will assume that inverting these functions requires negligible effort.

Observation 2. The quasigroup operation $Q(x, y)$ of Edon-R is easily invertible as well, i.e. it is easy to find x if y and $Q(x, y)$ are fixed. Same holds for y.

If, for example, we fix the values of y and $Q(x, y)$ then we get an equation of a type $\mu(x) = C$. From the observation 1, it follows that we can invert the quasigroup operation $Q(x, y)$ with negligible effort.

3 Free start second preimages and free start collisions for Edon-R

Definition. Let $f(x, y)$ be a compression function of an iterative Merkle-Damgard hash function $H(z)$, where x is the message, and y is the chaining value. The pairs (M^1, H^1) and (M^2, H^2) produce *a free start collision* if $f(M^1, H_1) = f(M^2, H_2)$.

Definition. Let $f(x, y)$ be a compression function of an iterative Merkle-Damgard hash function $H(z)$, where x is the message, and y is the chaining value. Let the message M_1 and the chaining value H^1 be fixed. The pair (M^2, H^2) produces *a free start second preimage* of (M^1, H^1) if $f(M^1, H^1) = f(M^2, H^2)$.

Let M^1 and H^1 produce the target hash value H^*. This pair generates internal blocks $(A^1_1, B^1_1), \ldots, (A^1_8, B^1_8) = H^*$. We will find another pair M^2, H^2 that generates internal blocks $(A^2_1, B^2_1), \ldots, (A^2_8, B^2_8)$, such that $(A^1_8, B^1_8) = (A^2_8, B^2_8) = H^*$, thus providing a free start second preimage. The main goal is to correct only H^2 such that this pair will produce the required hash value. This is done by step-by-step correction of the internal blocks A^2_i and B^2_i. We use the invertibility of the quasigroup operation in the case when two of the three values are fixed. Hence, sequentially, we define the internal blocks values. In the last two steps of the algorithm the exact values of H^2_1 and H^2_2 are found such that all previous equations hold. The algorithm steps are shown in Fig. 1 and work as follow:

Free start second preimage algorithm

Input $M^1 = (M^1_1, M^1_2), H^1 = (H^1_1, H^1_2)$

1. Generate $(A^1_1, B^1_1), \ldots, (A^1_8, B^1_8)$ for M^1, H^1.

2. Take any message $M_2 = (M_1^2, M_2^2)$, such that $M^2 \neq M^1$ in both blocks.
3. Find the value of A_6^2 from the equation $Q(\overline{M_1^2}, A_6^2) = A_7^1$. Note that $A_5^2 = A_6^2$.
4. Find the value of B_5^2 from the equation $Q(B_5^2, A_5^2) = B_6^1$. Note that $B_4^2 = B_5^2$.
5. Find the value of $A_1^2 = Q(\overline{M_2^2}, M_1^2)$ and $B_2^2 = Q(A_1^2, M_2^2)$. Note that $A_2^2 = A_1^2$ and $B_3^2 = B_2^2$.
6. Find the value of A_3^2 from the equation $Q(A_3^2, B_3^2) = B_4^2$. Note that $A_4^2 = A_3^2$.
7. Find the value of H_2^2 from the equation $Q(H_2^2, A_2^2) = A_3^2$. Note that $A_4^2 = A_3^2$.
8. Find the value of H_1^2 from the equation $Q(A_4^2, H_1^2) = A_5^2$.

Output M^2,H^2

The algorithm uses five inversions. Therefore, the second preimage can be found with negligible effort.

Remark. Notice that we can freely choose the colliding messages.

4 Free start preimages for Edon-R

Definition. Let $f(x, y)$ be a compression function of an iterative Merkle-Damgard hash function $H(z)$, where x is the message, and y is the chaining value. Let H^* be some hash value. The pair (M, H) is a free start preimage of H^* for f if $f(M, H) = H^*$.

Let $H^* = (H_1^*, H_2^*)$ is the target hash value (chaining value). We will find a pair M, H that generates internal blocks $(A_1, B_1), \dots, (A_8, B_8)$, such that $(A_8, B_8) = H^*$, hence produce a free start preimage. This is done by step-by-step fixing the values of the internal blocks, starting from the output block and ending with the values of the input chaining blocks.

Free start preimage algorithm

Input $H^* = (H_1^*, H_2^*)$

1. Take any message $M = (M_1, M_2)$
2. Fix the value of A_8 to H_1^* and B_8 to H_2^*. Note that $A_7 = A_8$.
3. Find the value of A_6 from the equation $Q(\overline{M_1}, A_6) = A_7$. Note that $A_5 = A_6$.
4. Find the value of B_7 from the equation $Q(A_7, B_7) = H_2^*$. Note that $B_6 = B_7$.
5. Find the value of B_5 from the equation $Q(A_5, B_5) = B_6$. Note that $B_4 = B_5$.
6. Find the value of $A_1 = Q(\overline{M_2}, M_1)$. Note that $A_2 = A_1$.
7. Find the value of $B_2 = Q(A_1, M_2)$. Note that $B_3 = B_2$.
8. Find the value of A_3 from the equation $Q(A_3, B_3) = B_4$. Note that $A_4 = A_3$.
9. Find the value of H_2 from the equation $Q(H_2, A_2) = A_3$.
10. Find the value of H_1 from the equation $Q(A_4, H_1) = A_5$.

Output M,H

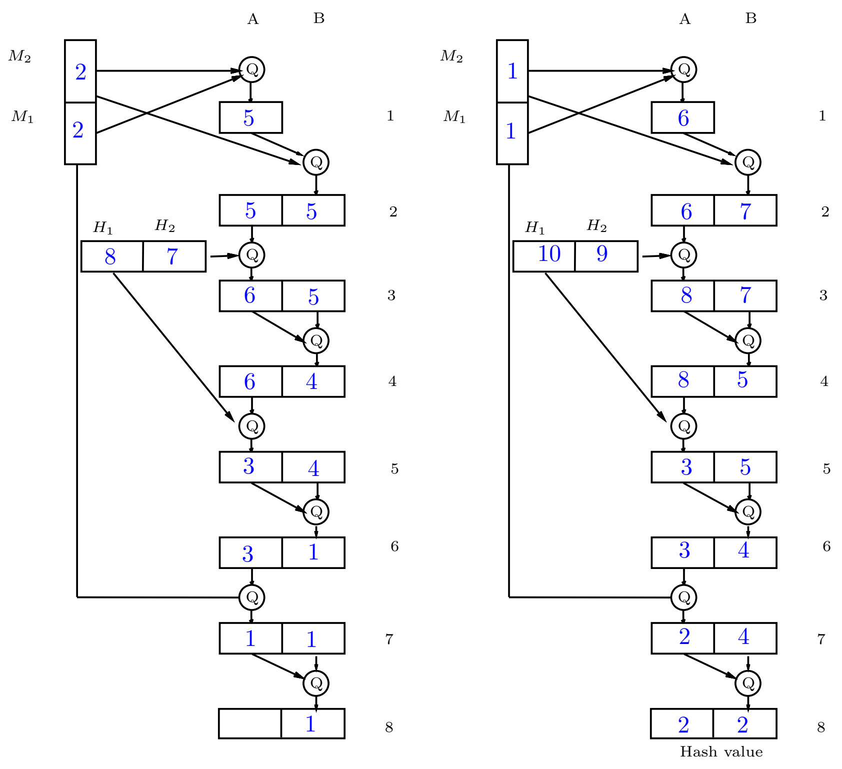

Figure 1: On the left, steps of the algorithm (numbers in the blocks) for finding free start second preimages. On the right, steps of the algorithm for finding free start preimages.

The algorithm uses six inversions. Therefore, the preimage can be found with negligible effort.

Remark. Notice that we can freely choose the preimage.

5 Preimage attack

The preimage attack is maintained using the meet-in-the-middle approach. The common procedure for Edon-n where n is the digest size[2] is defined as follows.

[2] The attack deals with 256-bit and 512-bit versions. The versions with truncated output can be attacked as well with the same complexity as corresponding non-truncated versions.

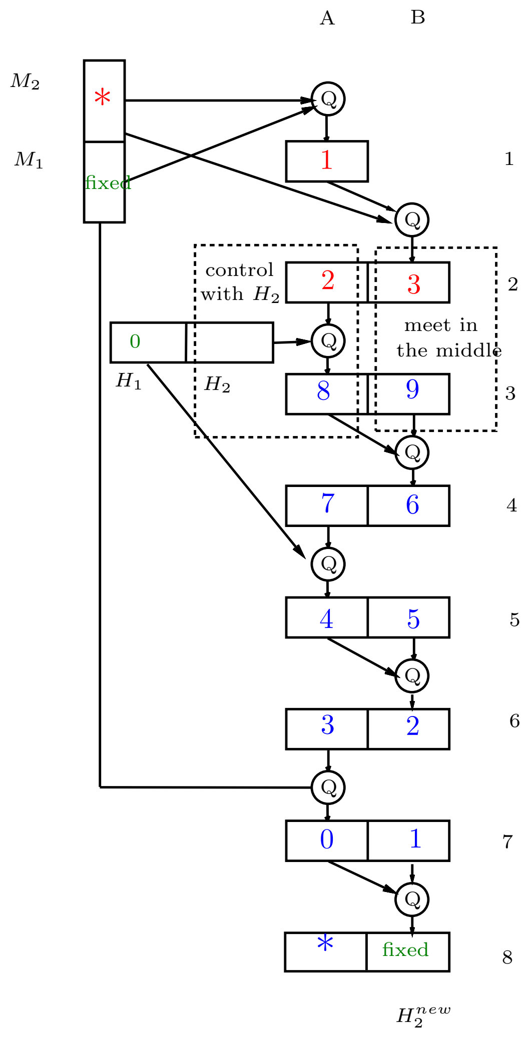

Figure 2: Meet-in-the-middle attack for partially inverting the compression function of Edon-R.

Roughly, we prepare a set S_1 of intermediate hash values that are produced from the initial value, and a set S_2 of intermediate values that are produced from the given final hash value by partially inverting the compression function. If these sets intersect we can build a message that leads to the given hash.

Since the intermediate value is twice as big as the hash digest we need to put a constraint on the intermediate states: we keep only states with $H_1 = 0$. Surprisingly, this filtering can be maintained in both directions with less than 2^n additional effort thus making the attack possible. Let us explain the attack in details.

Fixing H_1 in forward direction. We need only one message block to get a desired H_1^{new}. We want it to be 0, and both the initial value blocks are fixed as well. We claim that for each M_1 we can find M_2 such that this message input and the initial value blocks will produced a zero value in H_1^{new}.

Indeed, let M_1 be assigned with some random value. Then we obtain the value of A_6 since $A_7 = H_1^{\text{new}} = 0$ and the function Q is invertible. Then we consecutively obtain the values of A_5, A_4, A_3, A_2, and A_1 (keep in mind that the initial chaining value is fixed). Given A_1 and M_1, we derive M_2 by inverting the first application of Q. Then we obtain all B's and thus a pair $(H_1^{\text{new}} = 0, H_2^{\text{new}})$.

We repeat this step $2^{2n/3}$ times with different values of M_1 and get $2^{2n/3}$ valid pairs $(0, H_2^{\text{new}})$. We store the pairs with the messages.

Fixing H_1 in backward direction. Again, we need only one step (one message block) to get a pair of form $(0, H_2)$ from a given hash value $H = H_2^{\text{new}}$. However, this step is time-consuming. We get the desired pairs with another meet-in-the-middle procedure.

First, we assign M_1 with some predefined value m. Then we assign A_8 with some random value and consecutively obtain the values of the following internal variables (in this order): A_7, B_7, B_6, A_6 (using M_1), A_5, B_5, B_4, A_4, A_3, B_3. We repeat this step $2^{n/3}$ times for different values of A_8 and store pairs (A_8, B_3).

Now we assign M_2 with some random value[3] and obtain the values of A_1, A_2, and B_2 using the value of M_1. We repeat this step $2^{2n/3}$ times and store pairs (M_2, B_2). Note that H_2 is left undefined in both steps.

To build a correct execution it is enough to find two pairs (one from each set) such that $B_2 = B_3$. If B_2 and B_3 collide then we compute all the other internal variables without contradiction. Finally, we obtain H_2 from A_3 and A_2.

Since blocks have n bits we get about $2^{2n/3+2n/3-n} = 2^{n/3}$ colliding pairs and thus about $2^{n/3}$ valid pairs $(H_1 = 0, H_2)$.

Meet-in-the-middle. We have prepared two sets of pairs $(0, H_2)$: first one of size $2^{2n/3}$ and the second one of size $2^{n/3}$. With high probability, there are two elements from different sets colliding on H_2, which gives us a correct preimage.

Complexity. The attack needs $2^{2n/3}$ computations (compression function queries) and roughly the same amount of memory. A trivial time-memory tradeoff can be applied and would require about $2^{2n/3+k}$ operations and $2^{2n/3-k}$ memory.

References

1. Gligoroski, D., Ødegård, R.S., Mihova, M., Knapskog, S.J., Kocarev, L., Drápal, A.:Cryptographic Hash Function Edon-R. `http://people.item.ntnu.no/~danilog/Hash/Edon-R/Supporting_Documentation/EdonRDocumentation.pdf`

[3] The value is not truly random: 65 bits of the last message block are reserved for padding. However, we need to vary only $2n/3 < n - 65$ bits — see below.

A Free Start (Second) Preimage Examples for Edon-256

Table 1. Second Preimage Example.

$M1$	"This is a random message input."
IV_1	As defined in the specifications
$M2$	"The second input is different from the first."
IV_2	5450f88d a7ded66c 42948f72 1c9f9491 4935c383 f9fbfb82 7a529df2 3d978880
	485cdd0c 348ce98e 23a6c2f1 75ee5285 7f34305e 4000567a 4fd6f191 783665e2
$H_1 = H_2$	7260b283 5f1ab7cf e2140e05 3fe875e0 cb811d29 24a8950e d7000b9e c9cbb78b

Table 2. Preimage Example.

M	"NIST Competition"
IV	1a1b741 e81f784c e50b1b4a c580098f ccce0de8 fd904596 c8ac1760 625c09b4
H	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000