Free-start attacks on NaSHA

Ivica Nikolić and Dmitry Khovratovich

University of Luxembourg

Abstract. We present a free-start collision and a free-start preimage attack on NaSHA. The attacks exploit the fact that when the state, obtained after the linear transformation, is only partially fixed then the quasigroup operations are fully determined. The free-start collision attack requires 2^{32} computations for all digests. The free-start preimage attack requires around $2^{n/2}$ for NaSHA-n[1]. The attacks show a weakness in the compression function of NaSHA, yet they do not contradict the NIST security requirements.

1 Description of NaSHA

The hash family NaSHA[1] is based on the Merkle-Damgard construction with a double pipe chaining value. The compression function f of NaSHA takes two inputs: the message and the previous chaining value. The internal state is a quadruple pipe state. This state can be represented as 16 64-bit words for NaSHA-256, and 32 words for NaSHA-512. The compression function $f(M,H)$ can be represented as $f(M,H) = \mathcal{MT}(LinTr(t(M,H)))$, where $LinTr$ is a linear transformation, and $\mathcal{MT} = \mathcal{RA} \circ \mathcal{A}$ is an unbalanced Feistel network. The exact definition of the underlying transformations can be found in [1].

2 Free-start collisions

We will present a free-start collision attack on NaSHA with differences in the message input and the chaining value.
First, we will explain how to launch a truncated differential attack only on $\mathcal{MT}$ part (the differential is presented in Fig. 1). Then, we will extend this attack for the whole compression function.
Let us denote $\tilde{L} = (L_1, L_1, \ldots, L_{2q})$ the internal state of NaSHA after the linear transformation $LinTr$. Let two states $\tilde{L^1}, \tilde{L^2}$ differ only in the first two words L_1, L_2, such that $L_1^1 + L_2^1 = L_1^2 + L_2^2 = C$ ($L_i^1 = L_i^2, i > 2$). Then, for both of the states the quasigroup transformations $\mathcal{A}$ and $\mathcal{RA}$ are identically determined. Their leaders are also equal.
Now, let us take a look at the Feistel network used in NaSHA. In $\mathcal{A}$, first is updated L_1. Let $K_1 = F(l_1, L_1)$ be the updated value of this word. Since there is a difference (between $\tilde{L_1}$ and $\tilde{L_2}$) in L_1 it follows that K_1^1 and K_1^2 also differ. Next is updated L_2. Let $K_2 = F(K_1, L_2)$. We

[1] Truncated digests require the same effort as the non-truncated.

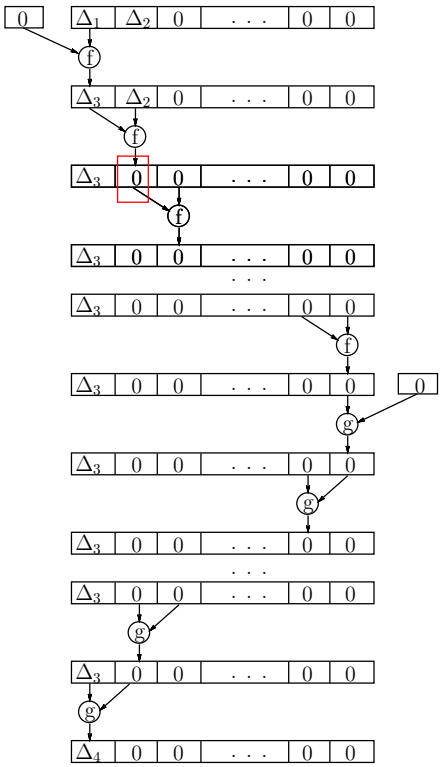

Figure 1: Truncated differential for the $\mathcal{MT}$ transformation used in the free-start collision attack.

require zero difference in K_2, i.e. $F(K_1^1, L_2^1) = F(K_1^2, L_2^2)$. Note that this is possible because there is a difference in both of the input arguments. The condition can be rewritten as:

$$F(F(l_1, L_1^1), C - L_1^1) = F(F(l_1, L_1^2), C - L_1^2)$$

This means that we need a collision for the function $F(F(l_1, X), C - X)$. By the birthday paradox, this collision can be produced with 2^{32} computations. After we have obtained a zero difference in K_2, then all of the following updates in $\mathcal{A}$ produce also zero difference since the input words and the quasigroup operations are equal. Now, let us take a look

at $\mathcal{RA}$. Similarly, all the updates produce zero difference. Only the last update, produces some difference in the first word. But this word is not taken in the output chaining value. Therefore, the presented differential, produces collisions for the $\mathcal{MT}$ part.
Now, let us try to extend the attack for the whole compression function. Let $\tilde{L_1}, \tilde{L_2}$ be two states that produce collision for $\mathcal{MT}$. Then, since the linear transformation is easily invertible, for each of the state it is possible to find the starting initial state value (chaining value and the message input). Hence, we can obtained a free-start collision for the whole compression function. The attack complexity is 2^{32}.
Note that if, when inverting these two states, we have obtained initial states with differences only in the message words, this would mean a real collision for the compression function. Yet, for the linear transformation, specified in NaSHA, this type of scenario is not possible for the differential that we have presented.

3 Free-start preimages

We will show how to attack NaSHA-256. Similar technique can be used for attacking NaSHA-512. Our attack strategy is the following. Again we start with an internal state $\tilde{L} = (L_1, L_1, \ldots, L_{16})$ obtained after the linear transformation. There we fix the sums L_1+L_2, $L_3+L_4, \ldots, L_{15}+L_{16}$. Thus, the quasigroup operations and the leaders are fully determined. Then, depending on the target hash value H^*, we show how to find the exact values of the words $L_1, L_2, \ldots, L_{16}$. At the end, we invert the linear transformation and obtain an initial state, hence a free start preimage. Before presenting the attack in details let us first deal with the quasigroup operations.

Quasigroup transformations. Each of the quasigroup transformations $\mathcal{A}$ and $\mathcal{RA}$ used in NaSHA consists of 16 applications of quasigroup operation $*$. Let $F(x, y) = (x + y) * y$ be the operation used in $\mathcal{A}$, and $G(x, y) = x * (x + y)$ the one used in $\mathcal{RA}$. Further in the attack, we will need to invert these two functions. If either x or y is fixed, then inverting the functions is trivial: we have to try all 2^{64} possible inputs. Hence, we can assume that one inversion costs 2^{64} computations. Another way of inverting the functions is using precomputed tables with all possible input-output values. Then, inverting costs only 128 computations (binary search among 2^{128} values), but the memory cost is 2^{128}. The functions are invertible with high probability as stated in [1, Proposition 4].

Now we can present the attack. Let $\tilde{L} = (L_1, L_1, \ldots, L_{16})$ be the internal state after the linear transformation, and let

$$L_1 + L_2 = L_3 + L_4 = \ldots = L_{15} + L_{16} = C^2 \tag{1}$$

Note that we fix only the sums but not the exact values of the words. With superscript i we will denote the state after the i-th update of $\mathcal{MT}$.

[2] Different constants can be used for different pairs.

For example, the initial state is $\tilde{L}^0$, the state after $\mathcal{A}$ is $\tilde{L}^{16}$, and the final state, after the whole $\mathcal{MT}$ is $\tilde{L}^{32}$. Let $H^* = (H_1, H_2, H_3, H_8)$ be the target hash value. We will show how to fix, sequentially, all the words in $\tilde{L}^0, \tilde{L}^1, \ldots, \tilde{L}^{16}$. Note that we deal with unbalanced Fiestel network, the words of two sequential states $\tilde{L}^{i-1}$, $\tilde{L}^i$ differ only in L_i. Below is the algorithm that finds the state $\tilde{L}^0$ from any target hash value (see Fig. 2).

1. Fix $L_4^{29} = H_1, L_8^{25} = H_2, L_{12}^{21} = H_3, L_{16}^{17} = H_8$
2. Find L_{16}^{16} from $G(L_{16}^{17}, l_2) = L_{16}^{16}$
3. Take random value for L_{16}^0. Then $L_{15}^0 = C - L_{16}^0$
4. Find L_{15}^{15} from $F(L_{15}^{15}, L_{16}^{15}) = L_{16}^{16}$.
5. Find L_{18}^{15} from $G(L_{15}^{17}, L_{16}^{17}) = G(L_{15}^{15}, L_{16}^{17}) = L_{18}^{15}$
6. Find L_{14}^{14} from $F(L_{14}^{14}, L_{15}^{14}) = L_{15}^{15}$
7. Find L_{14}^{19} from $G(L_{14}^{18}, L_{15}^{18}) = G(L_{14}^{14}, L_{15}^{18}) = L_{14}^{19}$
8. Take random value for L_{14}^0. Then $L_{13}^0 = C - L_{14}^0$
9. Find L_{13}^{13} from $F(L_{13}^{13}, L_{14}^{13}) = F(L_{13}^{13}, L_{14}^0) = L_{14}^{14}$.
10. Find L_{13}^{20} from $G(L_{13}^{19}, L_{14}^{19}) = G(L_{13}^{13}, L_{14}^{19}) = L_{13}^{20}$
11. Find L_{12}^{20} from $G(L_{12}^{20}, L_{13}^{20}) = L_{12}^{21}$. Find L_{12}^{12} from $F(L_{12}^{12}, L_{13}^{12}) = L_{13}^{13}$. If $L_{12}^{20} \neq L_{12}^{12}$ go to step 8.
12. Take random value for L_{12}^0. Then $L_{11}^0 = C - L_{12}^0$
13. Find L_{11}^{11} from $F(L_{11}^{11}, L_{12}^{11}) = F(L_{11}^{11}, L_{12}^0) = L_{12}^{12}$.
14. Find L_{11}^{22} from $G(L_{11}^{21}, L_{12}^{21}) = G(L_{11}^{11}, L_{12}^{21}) = L_{12}^{22}$.
15. Find L_{10}^{10} from $F(L_{10}^{10}, L_{11}^{10}) = L_{11}^{11}$.
16. Find L_{10}^{23} from $G(L_{10}^{22}, L_{11}^{22}) = G(L_{10}^{10}, L_{11}^{22}) = L_{10}^{23}$.
17. Take random value for L_{10}^0. Then $L_9^0 = C - L_{10}^0$.
18. Find L_9^9 from $F(L_9^9, L_{10}^9) = F(L_9^9, L_{10}^0) = L_{10}^{10}$.
19. Find L_9^{24} from $G(L_9^{23}, L_{10}^2 3) = G(L_9^9, L_{10}^2 3) = L_9^{24}$.
20. Find L_8^8 from $F(L_8^8, L_9^8) = F(L_9^9)$. Find L_8^{24} from $G(L_8^{24}, L_9^{24}) = L_8^{25}$. If $L_8^{24} \neq L_8^8$ go to step 17.
21. Take random value for L_8^0. Then $L_7^0 = C - L_8^0$.
22. Find L_7^7 from $F(L_7^7, L_8^7) = L_8^8$.
23. Find L_7^{26} from $G(L_7^{25}, L_8^{25}) = G(L_7^7, L_8^{25}) = L_7^{26}$.
24. Find L_6^6 from $F(L_6^6, L_7^6) = L_7^7$.
25. Find L_6^{27} from $G(L_6^{26}, L_7^{26}) = G(L_6^6, L_7^{26}) = L_6^{27}$.
26. Take random value for L_6^0. Then $L_5^0 = C - L_6^0$.
27. Find L_5^5 from $F(L_5^5, L_6^5) = L_6^6$.
28. Find L_5^{28} from $G(L_5^{27}, L_6^{27}) = G(L_5^5, L_6^{27}) = L_5^{28}$.
29. Find L_4^4 from $F(L_4^4, L_5^4) = L_5^5$. Find L_4^{28} from $G(L_4^{28}, L_5^{25}) = L_4^{29}$. If $L_4^{28} \neq L_4^4$ go to step 26.
30. Take random value for L_4^0. Then $L_3^0 = C - L_4^0$.
31. Find L_3^3 from $F(L_3^3, L_4^3) = F(L_3^3, L_4^0) = L_4^4$.
32. Find L_3^{30} from $G(L_3^{29}, L_4^{29}) = G(L_3^3, L_4^{29}) = L_3^{30}$.
33. Find L_2^2 from $F(L_2^2, L_3^2) = F(L_2^2, L_3^0) = L_3^3$.
34. Find L_2^{31} from $G(L_2^{30}, L_3^{30}) = G(L_2^2, L_3^{30}) = L_2^{31}$.
35. Take random value for L_2^0. Then $L_1^0 = C - L_2^0$.
36. Find L_1^1 from $F(L_1^1, L_2^1) = F(L_1^1, L_2^0) = L_2^2$. If $F(l_1, L_1^0) \neq L_1^1$ go to step 35.

After we find the state $\tilde{L}^0$ we can easily invert the linear transformations and obtain the initial state and thus find the input message and the chaining value.

Now let us find the complexity of the whole preimage search. Note that the main algorithm can be divided into 4 parts. The first part are steps

1-11, the second part are steps 12-20, the third part are 21-26, and the last part are steps 30-36. We can assume that each part is a separate algorithm which takes some input (from the previous part), and *always* finds the right value for some of the state words $\tilde{L}^0$. Hence, when computing the complexity of the whole attack, we need to add the complexities of these four parts (rather than multiply). Each part has a loop which runs around 2^{64} times and uses some inversion of F and G. Hence, the complexity of each part is around 2^{128}. The complexity of the whole algorithm is 2^{130}. If a look-up table is used for inverting F and G then the algorithm would require 2^{66} computations and 2^{128} memory.

References

1. Smile Markovski, Aleksandra Mileva: NaSHA family of cryptographic hash funcions. `http://inf.ugd.edu.mk/images/stories/file/Mileva/part2b1.pdf`

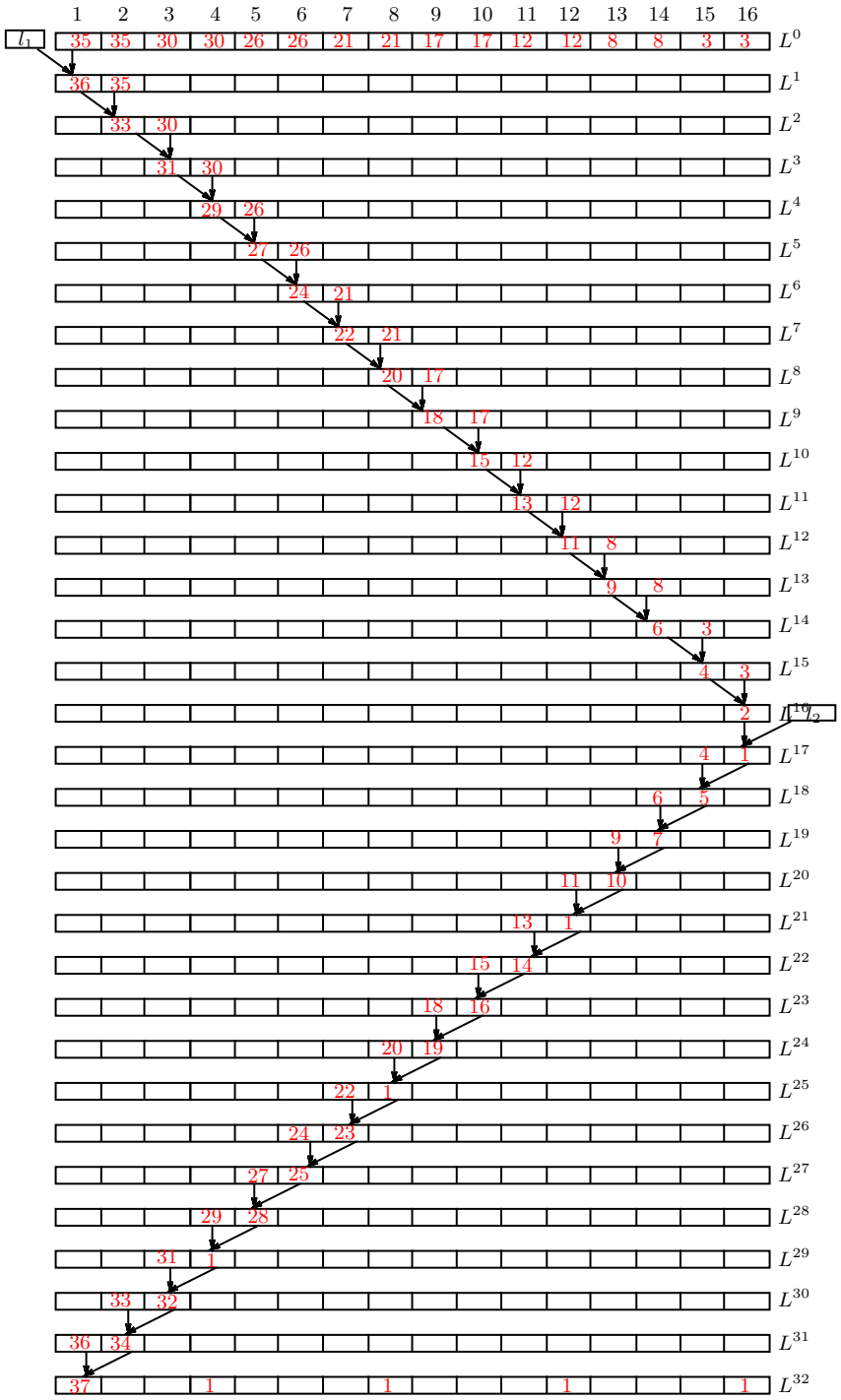

Figure 2: Steps of the free-start preimage finding algorithm.