Security Analysis of SIMD[*]

Charles Bouillaguet, Pierre-Alain Fouque, and Gaëtan Leurent

École Normale Supérieure – Département d'Informatique,
45 rue d'Ulm, 75230 Paris Cedex 05, France
{Charles.Bouillaguet,Gaetan.Leurent,Pierre-Alain.Fouque}@ens.fr

Abstract. This paper provides three important contributions to the security analysis of SIMD. First, we show a new free-start distinguisher based on symmetry relations. It allows to distinguish the compression function of SIMD from a random function with a single evaluation. However, we also show that this property is very hard to exploit to mount any attack on the hash function because of the mode of operation of the compression function. Essentially, if one can build a pair of symmetric states, the symmetry property can only be triggered once. Then, we show that a class of free-start distinguishers is not a threat to wide-pipe hash functions. In particular, this means that our distinguisher has a minimal impact on the security of the SIMD hash function, and we still have a security proof for the iterated hash function. Intuitively, the reason why this distinguisher does not weaken the function is that getting into a symmetric state is about as hard as finding a preimage. Finally, we study differential path in SIMD, and give an upper bound on the probability of related key differential paths. Our bound is in the order of $2^{-n/2}$ using very weak assumptions. Resistance to related key attacks is often overlooked, but it is very important for hash function designs.

Key words: SIMD, SHA-3, hash function, distinguisher, security proof with distinguishers.

1 Introduction

SIMD is a SHA-3 candidate designed by Leurent, Fouque and Bouillaguet [11]. Its main feature is a strong message expansion whose aim is to thwart differential attacks. In this paper we study the security of SIMD, and we introduce three new results.

In Section 2 we study its resistance against self-similarity attacks [4]. This class of attack is inspired by the complementation property of DES and includes symmetry based attacks. In the case of SIMD, we show that it is possible to exploit the symmetry of the design using special messages. This shows that the constants included in the message expansion of SIMD are not sufficient to prevent symmetry relations, and non-symmetric constants should be added in the last steps of the message expansion. In-depth study of this symmetry property shows that it is much weaker than symmetry properties in CubeHash [1,8] or Lesamnta [4]. More precisely, most symmetry properties can be used to generate many symmetric states out of a single state, but this is not the case for SIMD.

In Section 3, we show a proof of security for the mode of operation used in SIMD, the truncated prefix-free Merkle-Damgård, in the presence of some efficient distinguishers on the compression function. The class of distinguisher we consider includes the symmetry based distinguisher, and also includes differential paths with a non-zero chaining value difference. This shows that the properties of the compression function of SIMD found so far do not affect the security of the iterated hash function. This part is also of independent interest and applies to other wide-pipe hash functions.

[*]This is the full version of a paper published in SAC 2010.

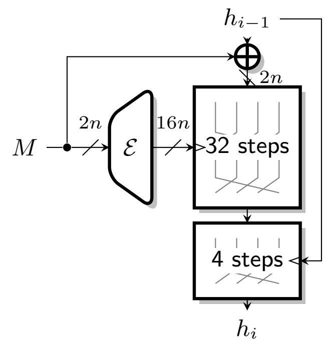

Fig. 1. SIMD modified Davies-Meyer mode

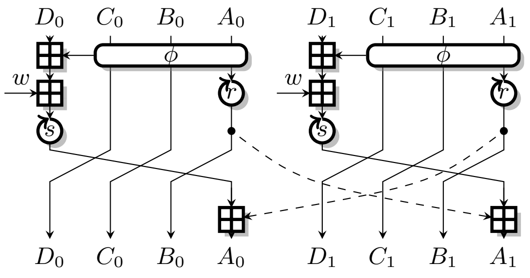

Fig. 2. SIMD compression rounds. There are 4 parallel Feistels in SIMD-256, and 8 parallel Feistels in SIMD-512.

In Section 4, we study differential attacks, and bound the probability of paths with a non-zero message difference, *i.e.*, related key attacks on the block cipher. We show an upper bound on such paths on the order of $2^{-n/2}$, and we argue that the best paths are probably much worse than this bound. We note that there are very few results known regarding resistance to related key attack for block ciphers. In particular, the differential properties of the AES have been extensively studied [16] but related key differential attacks have been shown recently [3]. In many hash function designs (in particular those based on the Davies-Meyer construction), related key attacks are a real concern and should be studied accordingly.

By combining the results of Section 3 and 4, we show that SIMD is resistant to differential cryptanalysis: a path with a non-zero difference in the chaining value input cannot be used to attack the hash function because it is wide-pipe, while a path a non-zero difference in the message can only have a low success probability.

Finally, in Section 5 we express our views on the security of SIMD.

1.1 Brief Description of SIMD

SIMD is built using a modified Davies-Meyer mode with a strong message expansion, as shown in Figure 1. The compression part is built from 4 parallel Feistel ladders (8 for SIMD-512) with 32-bit registers, and is shown in Figure 2. We can describe the step update function as:

$$D_j \leftarrow \left(D_j \boxplus W_j^{(i)} \boxplus \phi^{(i)}(A_j, B_j, C_j)\right)^{\lll s^{(i)}} \boxplus A_{p^{(i)}(j)}{}^{\lll r^{(i)}}$$

$$(A_j, B_j, C_j, D_j) \leftarrow (D_j, A_j^{\lll r^{(i)}}, B_j, C_j)$$

where j denotes the Feistel number, and i denotes the round number. A, B, C, and D are the four registers of the Feistel ladders, while $\phi^{(i)}$ is the Boolean function used at round i (which can be either IF or MAJ) and W is the expanded message. The parallel Feistels interact through the permutations $p^{(i)}$, which are built as $p^{(i)}(j) = j \oplus \alpha_i$, for some α_i. There are no explicit constants in the round function, but there are implicit constants in the message expansion.

The Message Expansion. The message expansion of SIMD is defined with the following operations:

1. Use a NTT transform (which is the same as a FFT over a finite field) to double the size of the message(the message bytes are interpreted as elements of $\mathbb{F}_{257}$). The NTT is actually used as a Reed-Solomon code.

Table 1. Full Permutations for SIMD-256. The y_i's are the output of the NTT, and $I_C(x, y)$ is $((C \boxtimes x) \bmod 2^{16}) + 2^{16}((C \boxtimes y) \bmod 2^{16})$

i	$W_0^{(i)}$	$W_1^{(i)}$	$W_2^{(i)}$	$W_3^{(i)}$
0	$I_{185}(y_{32}, y_{33})$	$I_{185}(y_{34}, y_{35})$	$I_{185}(y_{36}, y_{37})$	$I_{185}(y_{38}, y_{39})$
1	$I_{185}(y_{48}, y_{49})$	$I_{185}(y_{50}, y_{51})$	$I_{185}(y_{52}, y_{53})$	$I_{185}(y_{54}, y_{55})$
2	$I_{185}(y_0, y_1)$	$I_{185}(y_2, y_3)$	$I_{185}(y_4, y_5)$	$I_{185}(y_6, y_7)$
3	$I_{185}(y_{16}, y_{17})$	$I_{185}(y_{18}, y_{19})$	$I_{185}(y_{20}, y_{21})$	$I_{185}(y_{22}, y_{23})$
4	$I_{185}(y_{56}, y_{57})$	$I_{185}(y_{58}, y_{59})$	$I_{185}(y_{60}, y_{61})$	$I_{185}(y_{62}, y_{63})$
5	$I_{185}(y_{40}, y_{41})$	$I_{185}(y_{42}, y_{43})$	$I_{185}(y_{44}, y_{45})$	$I_{185}(y_{46}, y_{47})$
6	$I_{185}(y_{24}, y_{25})$	$I_{185}(y_{26}, y_{27})$	$I_{185}(y_{28}, y_{29})$	$I_{185}(y_{30}, y_{31})$
7	$I_{185}(y_8, y_9)$	$I_{185}(y_{10}, y_{11})$	$I_{185}(y_{12}, y_{13})$	$I_{185}(y_{14}, y_{15})$
8	$I_{185}(y_{120}, y_{121})$	$I_{185}(y_{122}, y_{123})$	$I_{185}(y_{124}, y_{125})$	$I_{185}(y_{126}, y_{127})$
9	$I_{185}(y_{88}, y_{89})$	$I_{185}(y_{90}, y_{91})$	$I_{185}(y_{92}, y_{93})$	$I_{185}(y_{94}, y_{95})$
10	$I_{185}(y_{96}, y_{97})$	$I_{185}(y_{98}, y_{99})$	$I_{185}(y_{100}, y_{101})$	$I_{185}(y_{102}, y_{103})$
11	$I_{185}(y_{64}, y_{65})$	$I_{185}(y_{66}, y_{67})$	$I_{185}(y_{68}, y_{69})$	$I_{185}(y_{70}, y_{71})$
12	$I_{185}(y_{72}, y_{73})$	$I_{185}(y_{74}, y_{75})$	$I_{185}(y_{76}, y_{77})$	$I_{185}(y_{78}, y_{79})$
13	$I_{185}(y_{104}, y_{105})$	$I_{185}(y_{106}, y_{107})$	$I_{185}(y_{108}, y_{109})$	$I_{185}(y_{110}, y_{111})$
14	$I_{185}(y_{80}, y_{81})$	$I_{185}(y_{82}, y_{83})$	$I_{185}(y_{84}, y_{85})$	$I_{185}(y_{86}, y_{87})$
15	$I_{185}(y_{112}, y_{113})$	$I_{185}(y_{114}, y_{115})$	$I_{185}(y_{116}, y_{117})$	$I_{185}(y_{118}, y_{119})$
16	$I_{233}(y_8, y_{72})$	$I_{233}(y_{10}, y_{74})$	$I_{233}(y_{12}, y_{76})$	$I_{233}(y_{14}, y_{78})$
17	$I_{233}(y_{16}, y_{80})$	$I_{233}(y_{18}, y_{82})$	$I_{233}(y_{20}, y_{84})$	$I_{233}(y_{22}, y_{86})$
18	$I_{233}(y_{56}, y_{120})$	$I_{233}(y_{58}, y_{122})$	$I_{233}(y_{60}, y_{124})$	$I_{233}(y_{62}, y_{126})$
19	$I_{233}(y_{32}, y_{96})$	$I_{233}(y_{34}, y_{98})$	$I_{233}(y_{36}, y_{100})$	$I_{233}(y_{38}, y_{102})$
20	$I_{233}(y_{48}, y_{112})$	$I_{233}(y_{50}, y_{114})$	$I_{233}(y_{52}, y_{116})$	$I_{233}(y_{54}, y_{118})$
21	$I_{233}(y_{40}, y_{104})$	$I_{233}(y_{42}, y_{106})$	$I_{233}(y_{44}, y_{108})$	$I_{233}(y_{46}, y_{110})$
22	$I_{233}(y_0, y_{64})$	$I_{233}(y_2, y_{66})$	$I_{233}(y_4, y_{68})$	$I_{233}(y_6, y_{70})$
23	$I_{233}(y_{24}, y_{88})$	$I_{233}(y_{26}, y_{90})$	$I_{233}(y_{28}, y_{92})$	$I_{233}(y_{30}, y_{94})$
24	$I_{233}(y_{49}, y_{113})$	$I_{233}(y_{51}, y_{115})$	$I_{233}(y_{53}, y_{117})$	$I_{233}(y_{55}, y_{119})$
25	$I_{233}(y_1, y_{65})$	$I_{233}(y_3, y_{67})$	$I_{233}(y_5, y_{69})$	$I_{233}(y_7, y_{71})$
26	$I_{233}(y_9, y_{73})$	$I_{233}(y_{11}, y_{75})$	$I_{233}(y_{13}, y_{77})$	$I_{233}(y_{15}, y_{79})$
27	$I_{233}(y_{57}, y_{121})$	$I_{233}(y_{59}, y_{123})$	$I_{233}(y_{61}, y_{125})$	$I_{233}(y_{63}, y_{127})$
28	$I_{233}(y_{25}, y_{89})$	$I_{233}(y_{27}, y_{91})$	$I_{233}(y_{29}, y_{93})$	$I_{233}(y_{31}, y_{95})$
29	$I_{233}(y_{41}, y_{105})$	$I_{233}(y_{43}, y_{107})$	$I_{233}(y_{45}, y_{109})$	$I_{233}(y_{47}, y_{111})$
30	$I_{233}(y_{33}, y_{97})$	$I_{233}(y_{35}, y_{99})$	$I_{233}(y_{37}, y_{101})$	$I_{233}(y_{39}, y_{103})$
31	$I_{233}(y_{17}, y_{81})$	$I_{233}(y_{19}, y_{83})$	$I_{233}(y_{21}, y_{85})$	$I_{233}(y_{23}, y_{87})$

2. Make two copies of the NTT output.
3. The first copy is multiplied by 185, while the second copy is multiplied by 233. This step also doubles the size of the message, as the output are 16-bit words.
4. Permute the 16-bit words and pack them into 32-bit words. Table 1 shows how the packing is done for SIMD-256.

Constants are added in the NTT layer, and make it an affine code instead of a linear one. They avoid special expanded messages such as the all-zero message. For more details, see the specification of SIMD [11].

1.2 Previous Cryptanalysis Results

As far as we know, the following results have been found on SIMD:

– In [9], Gauravaram and Bagheri showed that the modified Davies-Meyer construction used in SIMD allows to find partial fixed-points (this is a weaker version of Davies-Meyer's fixed-points).

There is no easy way to find full fixed-points as in the original Davies-Meyer construction, but those partial fixed-points give an easy distinguisher of the compression function. Just like the fixed-points of Davies-Meyer, this property does not affect the security of a wide-pipe hash function, and the mode can be proven secure under the assumption that the block cipher is ideal [7].
- In [14], Mendel and Nad showed a differential path with probability 2^{-507} for the compression function of the round-1 version of SIMD-512. They used it to make a distinguishing attack on the compression function with complexity 2^{427}, using IV/message modifications. In this path, no difference is introduced in the message, but a specific difference $\varDelta_{\text{in}}$ in the chaining value can go to a difference $\varDelta_{\text{out}}$. Because of the need to control the chaining value difference, this path cannot be used to attack the iterated hash function. In Section 3, we show that even if there of a path with probability one, we only loose a factor 2 in the indifferentiability proof. However, this path was using some unwanted properties of the permutations used in the compression function, and it was decided to remove those properties by tweaking the design for the second round of the SHA-3 competition [12].
- More recently, in [19] Yu and Wang studied differential paths for the round-2 version of SIMD. They describe near-collisions in reduced versions of the compression function (20 steps for SIMD-256 and 24 steps for SIMD-512) and build a differential path with probability 2^{-897} for the full compression function of SIMD-512. This path can be used to build a distinguisher with complexity 2^{398}, yielding pair of inputs and outputs with a fixed difference. Like the previous result, this work uses a difference in the chaining value and no difference in the message. For this reason it does not threaten the iterated hash function. It should be noted that for this distinguisher, the attacker needs to choose both input chaining values, and not only the difference between the chaining values (it is a *free-start* attack, while the attack on the round-1 version could be mounted as a *semi-free-start attack*). That makes it even less threatening to a wide-pipe design. Moreover we found several mistakes in the path described in the preprint of their work, which cast a doubt on the validity of the path.
- In [15], Nikolić *et al.* applied rotational cryptanalysis to the compression function of SIMD-512. They showed that 24 rounds can be distinguished from a random function with complexity 2^{497} if the constants are removed from the design. In the real design, they can only distinguish 12 rounds (out of 36) because of the non-linear message expansion. This is clearly not a threat for SIMD-512.

2 A Distinguisher for the Compression Function of SIMD

Our distinguisher is based on symmetries in the design, and follows the ideas of [4]. Symmetry based properties have already been found in several hash function designs, such as CubeHash [1,8] or Lesamnta [4]. We describe the distinguisher in the case of SIMD-256, but it applies similarly to SIMD-512.

2.1 Building the Symmetric Messages

The basic idea is to build a message so that the expanded message is symmetric. Then, if the internal state is also symmetric, the compression rounds preserve the symmetry. This can also be used with a pair of symmetric messages, and a pair of symmetric states.

The NTT layer of the message expansion is an affine transformation, therefore it is easy to find inputs that satisfy some affine conditions on the output. Since it only doubles the size of the input, we have enough degrees of freedom to force equalities between pairs of output. The next expansion step is a multiplication by a constant, and it will preserve equality relations.

If we look at the permutations used in the message expansion, they have the following property[1]: the NTT words used to build the message words $W_0^{(i)}, W_1^{(i)}, W_2^{(i)}, W_3^{(i)}$ are always of the form $(y_{k_1}, y_{k_2}), (y_{k_1+2}, y_{k_2+2}), (y_{k_1+4}, y_{k_2+4}), (y_{k_1+6}, y_{k_2+6})$ for some k_1 and k_2 (with $k_i = 0 \bmod 8$ or $k_i = 1 \bmod 8$). The full permutations are given in Table 1. Because of this property, if we have $y_i = y_{i\oplus 2}$ after the NTT, then we have $W_0^{(i)} = W_1^{(i)}$ and $W_2^{(i)} = W_3^{(i)}$. This allows us to build a symmetric message. An example of such a symmetric message is given in Appendix A.

More precisely, let us use the notation $\overleftrightarrow{\bullet}$ to denote this symmetry relation, and $\underleftrightarrow{\bullet}$ and $\underleftrightarrow{\overleftrightarrow{\bullet}}$ to denote the other two possible symmetries:

$$\begin{aligned}
\overleftrightarrow{(X_0, X_1, X_2, X_3)} &= (X_1, X_0, X_3, X_2)\\
\underleftrightarrow{(X_0, X_1, X_2, X_3)} &= (X_2, X_3, X_0, X_1)\\
\underleftrightarrow{\overleftrightarrow{(X_0, X_1, X_2, X_3)}} &= (X_3, X_2, X_1, X_0)
\end{aligned}$$

We now consider two messages M and M'. We use y to denote the NTT output for M, and y' to denote the NTT output for M'. The equality constraints on the NTT output that are necessary to build a pair of symmetric expanded messages are (we use $\mathcal{E}$ to denote the message expansion):

$$\begin{aligned}
y_i = y'_{i\oplus 2} &\Leftrightarrow \mathcal{E}(M) = \overleftrightarrow{\mathcal{E}(M')}\\
y_i = y'_{i\oplus 4} &\Leftrightarrow \mathcal{E}(M) = \underleftrightarrow{\mathcal{E}(M')}\\
y_i = y'_{i\oplus 6} &\Leftrightarrow \mathcal{E}(M) = \underleftrightarrow{\overleftrightarrow{\mathcal{E}(M')}}
\end{aligned}$$

In Appendix B we solve the linear systems involved, and we describe the sets of symmetric messages. For SIMD-256 we have the following results:

Symmetry class			# msg	# pairs
$\overleftrightarrow{\bullet}$	$y_i = y'_{i\oplus 2}$	$W_i = W'_{i\oplus 1}$	2^8	$256 \cdot 255$
$\underleftrightarrow{\bullet}$	$y_i = y'_{i\oplus 4}$	$W_i = W'_{i\oplus 2}$	2^{16}	$(256 \cdot 255)^2$
$\underleftrightarrow{\overleftrightarrow{\bullet}}$	$y_i = y'_{i\oplus 6}$	$W_i = W'_{i\oplus 3}$	2^8	$256 \cdot 255$

there are about 2^{16} symmetric messages, and less than 2^{32} symmetric pairs.

For SIMD-512 the results are:

Symmetry class		# msg	# pairs
$y_i = y'_{i\oplus 2}$	$W_i = W'_{i\oplus 1}$	2^8	$256 \cdot 255$
$y_i = y'_{i\oplus 4}$	$W_i = W'_{i\oplus 2}$	2^{16}	$(256 \cdot 255)^2$
$y_i = y'_{i\oplus 6}$	$W_i = W'_{i\oplus 3}$	2^8	$256 \cdot 255$
$y_i = y'_{i\oplus 8}$	$W_i = W'_{i\oplus 4}$	2^{32}	$(256 \cdot 255)^4$
$y_i = y'_{i\oplus 10}$	$W_i = W'_{i\oplus 5}$	2^8	$256 \cdot 255$
$y_i = y'_{i\oplus 12}$	$W_i = W'_{i\oplus 6}$	2^{16}	$(256 \cdot 255)^2$
$y_i = y'_{i\oplus 14}$	$W_i = W'_{i\oplus 7}$	2^8	$256 \cdot 255$

there are about 2^{32} symmetric messages, and less than 2^{64} symmetric pairs. An important property of these message classes is that they are all disjoint: it is not possible to use the intersection of two symmetry classes.

[1]This design choice was guided by implementation efficiency

2.2 Symmetry Property on the Compression Function

Let us consider a pair of symmetric messages for one of the symmetry relations (without loss of generality, we assume it's the $\overleftrightarrow{\bullet}$ symmetry): $\mathcal{E}(M') = \overleftrightarrow{\mathcal{E}(M)}$. We can take advantage of the symmetry of the Feistel part using those messages. If we have a pair of states $\mathcal{S}^{(i)}, \mathcal{S}'^{(i)}$ with $\mathcal{S}'^{(i)} = \overleftrightarrow{\mathcal{S}^{(i)}}$ and we compute one Feistel step with messages W and W' such that $W' = \overleftrightarrow{W}$, we obtain a new pair of states with $\mathcal{S}'^{(i+1)} = \overleftrightarrow{\mathcal{S}^{(i+1)}}$. The xor-based symmetry classes commute with the xor-based permutations $p^{(i)}$ used to mix the Feistels (and they are the only symmetry classes to do so).

Because the compression function is built using a modified Davies-Meyer mode (Figure 1), we need to start with H_{i-1} such that $H_{i-1} \oplus M$ is symmetric: $H'_{i-1} \oplus M' = \overleftrightarrow{H_{i-1} \oplus M}$. Then, in the feed-forward, H_{i-1} is used as the key to a few Feistel rounds, and since H_{i-1} is not symmetric, those rounds will break the symmetry. However, it turns out the symmetric messages are very sparse, so H_i will be almost symmetric, and the feed-forward will mostly preserve the symmetry of the outputs.

This gives a distinguisher on the compression function: an almost symmetric chaining value is transformed into a somewhat symmetric chaining value. See Appendix A for a concrete example.

The distinguisher can be used either with a pair of messages and chaining values with $\mathcal{E}(M') = \overleftrightarrow{\mathcal{E}(M)}$ and $H'_{i-1} \oplus M' = \overleftrightarrow{H_{i-1} \oplus M}$, or with a single chaining value and message, with $\mathcal{E}(M) = \overleftrightarrow{\mathcal{E}(M)}$ and $H_{i-1} \oplus M = \overleftrightarrow{H_{i-1} \oplus M}$.

2.3 Non-Ideality of the Compression Function

Here we define the bias of the compression function with the notations that will be used in Section 3. For each symmetric message M under a symmetry relation (denoted by $\overleftrightarrow{\bullet}$ without loss of generality), we have a first order relation between the inputs and output of the compression function:

$$\mathcal{R}_1^M(h, m, h') := \left(m = M \wedge h \oplus m = \overleftrightarrow{h \oplus m}\right) \Rightarrow P^{-1}(h', h) = \overleftrightarrow{P^{-1}(h', h)}$$

We use the feed-forward permutation P to define the relation, because it is tricky to describe exactly the somewhat symmetry of h' after the feed-forward. We have about 2^{16} such relations for SIMD-256 and about 2^{32} relations for SIMD-512. We can capture all of them in a single relation:

$$\mathcal{R}_1(h, m, h') := \bigwedge_M \mathcal{R}_1^M(h, m, h').$$

Similarly, for each symmetric message pair M, M', this gives a second order relation (there are about 2^{32} such relations for SIMD-256 and 2^{64} for SIMD-512):

$$\mathcal{R}_2^{M,M'}(h_1, m_1, h_2, m_2, h'_1, h'_2) :=$$
$$\left(m_1 = M \wedge m_2 = M' \wedge h_1 \oplus m_1 = \overleftrightarrow{h_2 \oplus m_2}\right) \Rightarrow P^{-1}(h'_1, h_1) = \overleftrightarrow{P^{-1}(h'_2, h_2)}$$

$$\mathcal{R}_2(h_1, m_1, h_2, m_2, h'_1, h'_2) := \bigwedge_{M,M'} \mathcal{R}_2^{M,M'}(h_1, m_1, h_2, m_2, h'_1, h'_2)$$

The corresponding weak states are:

$$\mathcal{W}_1^M := \{M \oplus x \mid x = \overleftrightarrow{x}\} \qquad \mathcal{W}_1 := \bigcup_M \mathcal{W}_1^M$$

$$\mathcal{W}_2^{M,M'} := \Big\{(h, \overleftrightarrow{h} \oplus M' \oplus \overleftrightarrow{M})\Big\} \qquad \mathcal{W}_2 := \bigcup_{M,M'} \mathcal{W}_2^{M,M'}$$

The study of the symmetry classes of SIMD, in Appendix B shows that:

$$|\mathcal{W}_1| = 2^{256} \cdot 256^2 + 2 \cdot 256 \approx 2^{256} \cdot 2^{16} \qquad \text{for SIMD-256}$$
$$|\mathcal{W}_1| = 2^{512} \cdot 256^4 + 2 \cdot 256^2 + 4 \cdot 256 \approx 2^{512} \cdot 2^{32} \qquad \text{for SIMD-512}$$
$$|\mathcal{W}_2| = 2^{512} \cdot ((256 \cdot 255)^2 + 2 \cdot 256 \cdot 255) < 2^{512} \cdot 2^{32} \qquad \text{for SIMD-256}$$
$$|\mathcal{W}_2| = 2^{1024} \cdot ((256 \cdot 255)^4 + 2 \cdot (256 \cdot 255)^2 + 4 \cdot 256 \cdot 255) < 2^{1024} \cdot 2^{64} \qquad \text{for SIMD-512}$$

Each chaining value can be used with less than 2^{32} related chaining values (less than 2^{64} for SIMD-512) and each such pair can be used with a single message.

2.4 Impact of the Symmetry-based Distinguisher

There are two main classes of attacks based on symmetric properties of the compression function. To attack the compression function, one can use the symmetry property to force the output of the compression function into a small subspace. This allows to find collisions in the compression function more efficiently than brute force, with the efficiency of this attack depending on the size of the symmetry classes. On the other hand, to attack the hash function, one can first try to reach a symmetric state using random messages, and then use symmetric messages to build a large set of symmetric states. To expand the set, the attacker will build a tree, starting with the symmetric state that was reached randomly. The degree and the depth of the tree can be limited depending on the symmetry property. In the case of SIMD, none of these attacks are effective for the following reasons:

- First, the modified Davies-Meyer mode of operation means that the compression function does not transform a symmetric state into a symmetric state, but it transforms an almost symmetric state into a somewhat symmetric state. We show in Appendix B that a "somewhat symmetric" output pair can only be used as an "almost symmetric" input pair with a very small probability. This prevents attacks based on building long chains of symmetric messages, like the attacks on CubeHash [1,8].
- Second, if a pair of almost symmetric states is reached, there is only a single message pair that can be used to reach a symmetric state in the Feistel rounds. This prevents attacks like the herding attack on Lesamnta [4], where one reaches a symmetric state and then uses a lot of different messages in order to explore the subset of symmetric outputs.
- Third, the final transformation of SIMD uses the message length as input. Therefore, the symmetry property can only be seen in the output of the hash function with messages of unrealistic length (almost 2^{512} bits for SIMD-256 and almost 2^{1024} bits for SIMD-512). Note that computing the hash of such a message is vastly more expensive than finding a preimage.
- Moreover the symmetry classes do not intersect. It is not possible to build a smaller symmetry classes in order to show collisions in the compression function, as was done for CubeHash [1,8]. Finding collisions in the compression function using the symmetry property costs $2^{n/2}$. It is more efficient than generic attacks on the compression function, but cannot be used to find collisions in the hash function faster than the birthday attack. We also note that the initial state of the SIMD hash function is not symmetric.

Table 2. Comparison of symmetry properties in several hash functions.

Function	Reach symm. state	Max. length	Max. degree	Free-start Collisions
Lesamnta-512	2^{256}	1	2^{256}	2^{128} (semi-free-start)
CubeHash (symm $C_1..C_7$)	2^{384}	∞	2^{128}	2^{32} (semi-free-start)
CubeHash (symm $C_8..C_{15}$)	2^{256}	∞	1	2^{64} (semi-free-start)
SIMD-512	2^{480}	1	1	2^{256}

To summarize, reaching a symmetric state in SIMD is far less interesting than reaching a symmetric state in CubeHash or in Lesamnta. Table 2 gives a comparison of the symmetry properties found in these functions.

Another very important factor is that SIMD is a wide-pipe design. Therefore reaching a symmetric state is about as hard a finding a preimage for the hash function. In the next section, we provide a formal proof that this distinguisher has only a small effect on the security of SIMD. We can prove that the hash function behaves as a random oracle under the assumption that the compression function is a weak perfect function having this symmetry property.

3 Free-start Distinguishers, Non-Ideal Compression Functions and Wide-Pipe Designs

In this section, we discuss the security of the prefix-free iteration of non-ideal compression functions. While our primary objective is to show that the distinguisher for the compression function of SIMD presented in Section 2 does not void the security proof of SIMD, the reasoning and the proof presented here are pretty general and could very well be adapted to other functions.

Let $\mathcal{H} = \{0,1\}^p$ denote the set of chaining values, $\mathcal{M} = \{0,1\}^m$ denote the set of message blocks, and $\mathcal{F}$ be the set of all functions $\mathcal{H} \times \mathcal{M} \to \mathcal{H}$. Let $F \in \mathcal{F}$ be a compression function taking as input an p-bit chaining value and an m-bit message block. A mode of operation for a hash function $H^{\cdot}$ combined with a compression function F yields a full hash function H^F.

Following [13,7], we rely on the notion of indifferentiability of systems to reduce the security of SIMD to that of its compression function. The usual way of establishing the soundness of a mode of operation $H^{\cdot}$ is to show that it is indifferentiable from a random oracle. This is done by constructing a simulator $\mathcal{S}$ such that any distinguisher $\mathcal{D}$ cannot tell apart (H^F, F) and $(RO, \mathcal{S})$ without a considerable effort, where RO is a variable-input-length random oracle (VIL-RO, for short). When this is established, it is shown in [13] that any cryptosystem making use of a VIL-RO is not less secure when the random oracle is replaced by the hash function H^F, where F is an ideal compression function (*i.e.*, a fixed-input-length random oracle, FIL-RO for short). Informally, if F is ideal (*i.e.*, has no special property that a random function would not have), then H^F is secure up to the level offered by the indifferentiability proof. More precisely, if $H^{\cdot}$ is $(t_{\mathcal{D}}, t_{\mathcal{S}}, q_{\mathcal{S}}, q_0, \varepsilon)$-indifferentiable from a VIL-RO when the compression function is assumed to be a FIL-RO, then this means that there exists a simulator running in time $t_{\mathcal{S}}$, such that any distinguisher running in time $t_{\mathcal{D}}$ and issuing at most $q_{\mathcal{S}}$ (resp. q_0) queries to the FIL-RO (resp. VIL-RO) has success probability at most ε.

A property of this methodology is that as soon as the compression function used in a hash function turns out to be non-ideal, then the security argument offered by the indifferentiability proof becomes vacuous. For instance, distinguishers exhibiting a "non-random" behavior of the compression function are usually advertised by their authors to nullify the security proof of the full hash function.

This problematic situation was first tackled by the designers of Shabal, who provided a security proof taking into account the existence of an efficient distinguisher on the internal permutation of their proposal [5]. We will follow their track and demonstrate that the security of SIMD can be proved despite the existence of an efficient distinguisher on its compression function.

The mode of operation of SIMD can be "concisely" described as being the wide-pipe prefix-free[2] iteration of the compression function. Let H^F therefore denote the *prefix-free* Merkle-Damgård iteration of F. Formally, $g : \{0,1\}^* \rightarrow \mathcal{M}^*$ is a *prefix-free encoding* if for all x, x', $g(x)$ is not a prefix of $g(x')$. The mode of operation $H^{\cdot}$ simply applies the Merkle-Damgård iteration of F to the prefix-free encoding of the message.

The original security argument was that if the internal state and the hash are both p-bit wide, then prefix-free Merkle-Damgård is indifferentiable from a random oracle up to about $2^{p/2}$ queries [7]. Theorem 1 below gives a formal statement of this result.

Theorem 1. *Prefix-Free Merkle-Damgård is $(t_{\mathcal{D}}, t_{\mathcal{S}}, q_S, q_O, \varepsilon)$-indifferentiable from a VIL-RO when the compression function is modeled by a FIL-RO, for any running time $t_{\mathcal{D}}$ of the distinguisher, and $t_{\mathcal{S}} = \mathcal{O}\left((q_O + \kappa \cdot q_S)^2\right)$ where κ is an upper-bound on the size of the queries sent to the VIL-RO. If $q = q_S + \kappa \cdot q_O + 1$, then the success probability of the distinguisher is upper-bounded by:*

$$\varepsilon = 8 \cdot \frac{q^2}{2^p}$$

In SIMD where the internal state is $2n$ bits, this ensures the indifferentiability of the whole function up to roughly 2^n queries (if H is indifferentiable up to q queries, then the composition of a truncation that truncates half of the output and of H is also secure up to q queries).

To restore the security argument damaged by the distinguisher, we will show that the prefix-free iteration of a non-ideal compression function is to some extent still indifferentiable from a VIL-RO.

3.1 Deterministic Distinguishers for the Compression Function

Let us consider a non-ideal compression function F.

- For instance, it may have *weak states*, that are such that querying F thereon with a well-chosen message block produces a "special" output allowing to distinguish F from random in one query. Known examples include for instance the symmetry on the compression function of Lesamnta [4], CubeHash [1,8], and SIMD (described in Section 2).
- But F can also have *bad second-order properties*, meaning that the output of F on correlated input states (with well-chosen message blocks) produces correlated outputs, allowing to distinguish F from random in two queries. A notable example of this property include the existence of differential paths with probability one in the compression function of Shabal [2]. Symmetry properties also give second order relations, which means that Lesamnta, CubeHash and SIMD have bad second-order properties as well.

Following the methodology introduced in [5], we model this situation by saying that there are two relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that:

$$\forall (h, m) \in \mathcal{H} \times \mathcal{M} : \qquad \mathcal{R}_1(h, m, F(h, m)) = 1$$
$$\forall (h_1, h_2, m_1, m_2) \in \mathcal{H}^2 \times \mathcal{M}^2 : \qquad \mathcal{R}_2(h_1, m_1, h_2, m_2, F(h_1, m_1), F(h_2, m_2)) = 1$$

[2]this is not explicitly stated in the submission document, but SIMD has a different finalization function that effectively acts as a prefix-free encoding.

We denote by $\mathcal{R}$ the relation formed by the union of $\mathcal{R}_1$ and $\mathcal{R}_2$, and we will denote by $\mathcal{F}[\mathcal{R}]$ the subset of $\mathcal{F}$ such that the above two equations hold. We require the relations to be efficiently checkable, *i.e.*, that given h, m and h', it is efficient to check whether $\mathcal{R}_1(h, m, h') = 1$. The relation can thus be used as an efficient distinguishing algorithm that tells $\mathcal{F}[\mathcal{R}]$ apart from $\mathcal{F}$.

A *weak state* is a state on which it is possible to falsify the relation $\mathcal{R}_1$. We formally define the set of weak states for $\mathcal{R}_1$ in the following way:

$$\mathcal{W} = \{h \in \mathcal{H} \mid \exists m, h' \in \mathcal{M} \times \mathcal{H} \text{ such that } \mathcal{R}_1(h, m, h') = 0\}$$

$\mathcal{W}$ should be a relatively small subset of $\mathcal{H}$ because the loss of security will be related to the size of $\mathcal{W}$. Moreover, we require that the IV is not in $\mathcal{W}$.

In the same vein, a *weak pair* is a pair of states on which it is possible to falsify the relation $\mathcal{R}_2$. We therefore define the set of *weak pairs* for $\mathcal{R}_2$ by an undirected graph $G_{\mathcal{R}_2} = (\mathcal{H}, \mathcal{WP})$, where $\mathcal{WP}$ is defined by:

$$\mathcal{WP} = \left\{h_1 \leftrightarrow h_2 \mid \exists m_1, m_2, h'_1, h'_2 \in \mathcal{M}^2 \times \mathcal{H}^2 \text{such that} \mathcal{R}_2(h_1, m_1, h_2, m_2, h'_1, h'_2) = 0\right\}$$

Similarly, $\mathcal{WP}$ should be a relatively small subset of $\mathcal{H}^2$ because the security loss will be related to the size of $\mathcal{WP}$. For the sake of expressing things conveniently, we define a variant of the same graph, $G'_{\mathcal{R}_2} = (\mathcal{H} \times \mathcal{M}, \mathcal{WP}')$, where $\mathcal{WP}'$ is defined by:

$$\mathcal{WP}' = \left\{(h_1, m_1) \leftrightarrow (h_2, m_2) \mid \exists h'_1, h'_2 \in \mathcal{H}^2 \text{such that} \mathcal{R}_2(h_1, m_1, h_2, m_2, h'_1, h'_2) = 0\right\}$$

To simplify the proof we also require that the connected component of $G'_{\mathcal{R}_2}$ have size at most two. This rules out some second-order relations, but it includes for instance the existence of a differential path with probability one with a non-zero difference in the input chaining value, as well as the symmetry in the compression function of SIMD or Lesamnta. We expect a similar result with larger connected components, but there will be a loss of security related to their size.

We also require the existence of *sampling algorithms* for $\mathcal{R}$, namely of two efficient algorithms **Sampler**$_1$ and **Sampler**$_2$ such that:

```
Sampler_1$(h, m)$
    $h' \xleftarrow{\$} \{f(h, m) \mid f \in \mathcal{F}[\mathcal{R}]\}$ ; return $h'$
Sampler_2$(h_1, m_1, h_2, m_2, h'_1)$
    $h'_2 \xleftarrow{\$} \{f(h_2, m_2) \mid f \in \mathcal{F}[\mathcal{R}] \text{ and } F(h_1, m_1) = h'_1\}$ ; return $h'_2$
```

Informally, the sampling algorithms should produce an output that looks as if it were produced by a random function constrained to conform to $\mathcal{R}$.

3.2 Adapting the Indifferentiability Proof to Non-Ideal Compression Functions

We now assume that the compression function is a public function chosen uniformly at random in $\mathcal{F}[\mathcal{R}]$, and for the sake of convenience we will call it a "biased FIL-RO". We show that the prefix-free iteration of biased FIL-RO is indifferentiable from a VIL-RO. In fact, we extend Theorem 1 to the case where the compression function is biased.

Theorem 2. *Prefix-Free Merkle-Damgård is $(t_\mathcal{D}, t_\mathcal{S}, q_S, q_O, \varepsilon)$-indifferentiable from a VIL-RO, when the compression function is modeled by a biased FIL-RO conforming to the relation $\mathcal{R}$, for any running time $t_\mathcal{D}$ of the distinguisher, and $t_\mathcal{S} = \mathcal{O}\left((q_O + \kappa \cdot q_S)^2\right)$ where κ is an upper-bound* *on the size of the queries sent to the VIL-RO. If $q = q_S + \kappa \cdot q_o + 1$, then the probability of success of the distinguisher is upper-bounded by:*

$$\varepsilon = 16 \cdot \frac{q^2}{2^p} + 4 \cdot |\mathcal{W}| \cdot \frac{q}{2^p} + 4 \cdot |\mathcal{WP}| \cdot \frac{q^2}{(2^p - q)^2}$$

The first term of the expression of ε is similar to the result given in Theorem 1, when the compression function is ideal (up to a factor two that could be avoided by making the argument slightly more involved). The two other terms reflect the fact that the compression function is biased. The relation induces a security loss if $|\mathcal{W}|$ is at least of order $2^{p/2}$, or if $|\mathcal{WP}|$ is at least of order 2^p. Informally, it seems possible to iterate compression functions having a relatively high bias in a secure way.

Application to Free-start Differential Attacks. Let us assume that the compression function is weak because of the existence of a good differential path with a non-zero difference in the input chaining value. Even if the probability of the differential path is 1, this has a very limited effect on the security of the hash function: this leads to $\mathcal{W} = \emptyset$ and $|\mathcal{WP}| = 2^{p-1}$. The advantage of the distinguisher is at most twice as high, compared to the iteration of an ideal FIL-RO.

Application to SIMD. In SIMD-256 (resp. SIMD-512), the internal state has $p = 512$ bits (resp. $p = 1024$ bits), and the distinguisher of Section 2 yields $|\mathcal{W}| = 2^{p/2+16}$, $|\mathcal{WP}| = 2^{p+32}$ (resp. $|\mathcal{W}| = 2^{p/2+32}$, $|\mathcal{WP}| = 2^{p+64}$). Therefore the advantage of any distinguisher in telling apart SIMD-256 from a VIL-RO with q queries is upper-bounded by:

$$\varepsilon = 16 \cdot \frac{q^2}{2^p} + 4 \cdot \frac{2^{p/2+16} \cdot q}{2^p} + 4 \cdot 2^{p+32} \cdot \frac{q^2}{(2^p - q)^2}$$

SIMD-256 is then secure up to roughly 2^{256-16} queries (SIMD-512 is secure up to 2^{512-32} queries).

Application to Lesamnta. Lesamnta follows the prefix-free Merkle-Damgård mode of operation due to its special finalization function. An efficient distinguisher based on symmetries was shown in [4], with $|\mathcal{W}| = 2^{p/2}$ and $|\mathcal{WP}| = 2^{p-1}$. According to Theorem 2, the advantage of any distinguisher in telling apart Lesamnta-256 from a random oracle with q queries is upper-bounded by:

$$\varepsilon = 16 \cdot \frac{q^2}{2^p} + 4 \cdot \frac{2^{p/2} \cdot q}{2^p} + 4 \cdot 2^{p-1} \cdot \frac{q^2}{(2^p - q)^2} \approx 22 \cdot \frac{q}{2^{p/2}}$$

Note that since Lesamnta is a narrow-pipe design, we have $p = n$. Our result shows that Lesamnta remains secure against generic attacks up to the birthday bound. This is the best achievable proof for Lesamnta, since it does not behave as a good narrow-pipe hash function beyond that bound: a dedicated herding attack based on the symmetry property is shown in [4], with complexity $2^{n/2}$.

3.3 Proof Sketch of Theorem 2.

We give a sketch of the proof, while a more formal proof can be found in Appendix D. The proof is heavily based on the proof in the extended version of [7]. The heart of the proof is a *simulator* $\mathcal{S}$ which has oracle access to the VIL-RO, and whose task is to simulate a biased FIL-RO. The pseudo-code of the simulator is shown in Figure 3, but a few preliminary remarks are in order. The simulator maintains a log of the queries it has answered to. This knowledge is maintained

```
1: function SIMULATOR$(h, m)$
2:     if there exist a vertex $h' \in V$ and an edge $h \xrightarrow{m} h'$ in $E$ then
3:         return this $h'$
4:     else
5:         return FRESHVALUE$(h, m)$
6:     end if
7: end function

8: function FRESHVALUE$(h, m)$
9:     if there exist $(u, v) \leftrightarrow (h, m) \in G'_{\mathcal{R}_2}$ then $(\overline{h}, \overline{m}) \leftarrow (u, v)$
10:    if $IV \overset{M}{\underset{*}{\rightarrow}} \overline{h} \in \mathsf{Reach}$ then Swap $(h, m)$ and $(\overline{h}, \overline{m})$        ▷ (only if $\overline{h}$ is defined)
11:    if $IV \overset{M}{\underset{*}{\rightarrow}} h \in \mathsf{Reach}$ then
12:        if there exist $M'$ such that $M \| m = g(M')$ then
13:            $h' \leftarrow RO(M')$
14:        else
15:            $h' \overset{\$}{\leftarrow} \mathcal{H}$
16:        end if
17:        $\overline{h'} \leftarrow$ Sampler$_2$ $(h, m, \overline{h}, \overline{m}, h')$        ▷ (only if $\overline{h}$ is defined)
18:        if $h' \in \mathcal{W}$ or $h' \in V$ or $\mathsf{Reach} \cup \{h'\}$ covers an edge of $G_{\mathcal{R}_2}$ then Abort
19:        $\mathsf{Reach} \leftarrow \mathsf{Reach} \cup \left\{h \xrightarrow{m} h'\right\}$
20:    else
21:        $h' \leftarrow$ Sampler$_1(h, m)$
22:        $\overline{h}' \leftarrow$ Sampler$_2$ $(h, m, \overline{h}, \overline{m}, h')$        ▷ (only if $\overline{h}$ is defined)
23:    end if
24:    $V \leftarrow V \cup \{h, h', \overline{h}, \overline{h'}\}$        ▷ (only add $\overline{h}$ and $\overline{h'}$ if defined)
25:    $E \leftarrow E \cup \left\{h \xrightarrow{m} h', \overline{h} \xrightarrow{\overline{m}} \overline{h'}\right\}$        ▷ idem.
26:    return $h'$ (or $\overline{h'}$ if they were swapped in line 10)
27: end function
```

Fig. 3. Pseudo-code of the Simulator $\mathcal{S}_0$, with abort conditions

under the form of a graph $G = (V, E)$, where the set of vertices V is a subset of $\mathcal{H}$, and where the edges are labelled by message blocks from $\mathcal{M}$. The semantic of this graph is that there is an edge labelled by m between h and h' if the simulator let the distinguisher know that $f(h, m) = h'$. We will use the notation $h \xrightarrow{m} h'$ to say that there is an edge between h and h' labelled by m in G. Initially, the graph contains only a single vertex IV. The simulator also maintains a subset of V denoted by Reach, consisting of the vertices that are reachable from IV. It also associates to each vertex v in Reach an *ancestor* in Reach. This allows to efficiently reconstruct the sequence of message blocks that map IV on v, given $v \in \mathsf{Reach}$. We will note $IV \overset{M}{\underset{*}{\rightarrow}} v$ when there is such a path between IV and v. In the beginning, Reach only contains the IV.

Now, a *distinguisher* $\mathcal{D}$ interacts with either H^F and F (we say that it is in the "construction world"), or with RO (which is a VIL-RO) and $\mathcal{S}$ (and we say that it is in the "random oracle world"), and it has to tell in which world it is. More formally, $\mathcal{D}$ is a Turing machine which has two interfaces. It should output "1" if H^F and F are answering its oracle queries, or "0" if RO and $\mathcal{S}$ are. Our objective is to show that for all distinguisher $\mathcal{D}$ the following holds for a small ε:

$$\left|\mathbb{P}\left[\mathcal{D}^{H^F, F} = 1\right] - \mathbb{P}\left[\mathcal{D}^{RO, \mathcal{S}} = 1\right]\right| \leq \varepsilon$$

The main idea of the proof is that our simulator aborts as soon as a state (or pair of states) on which the relation could be falsified becomes reachable (that is to say, a state in $\mathcal{W}$ or a pair of states in $\mathcal{WP}$). Therefore we do not have to study exactly how much information is revealed by the relation. We use the sampling algorithms to simulate the weakness of the compression function, but the adversary can never compare the outputs of the samplers with the output of the VIL-RO, because that would cause the simulator to abort. Moreover, when queried on (h, m), the simulator looks for a symmetric query $(\overline{h}, \overline{m})$ so that the relation $\mathcal{R}_2$ could be falsified (*i.e.*, $(h, m) \leftrightarrow (\overline{h}, \overline{m}) \in G'_{\mathcal{R}_2}$). If such a symmetric query exists, the simulator computes both queries at the same time to ensure that they respect the relation $\mathcal{R}_2$.

The proof uses a hybrid argument through a sequence of games, which we summarize.

Game 1: The distinguisher is in the random oracle world. It has access to RO and $\mathcal{S}$.

Game 2: We introduce a dummy *relay algorithm* $\mathcal{T}$, which sits between the distinguisher and the RO. Given a random oracle query from the distinguisher, $\mathcal{T}$ just sends the query to RO, and transmits the answer of RO back to $\mathcal{D}$. This leaves the view of $\mathcal{D}$ unchanged.

Game 3: We modify the simulator $\mathcal{S}$, by making it *abort* in some cases, and report failure. The failure of $\mathcal{S}$ ensures that specific invariants of its internal data structures hold. Specifically, when queried on a reachable chaining value, $\mathcal{S}$ fails if its answer was already "known" by the distinguisher from a previous and different query. Thus, a (reachable) collision on the internal state, or the "connection" to some internal state already known would make $\mathcal{S}$ fail. Moreover, $\mathcal{S}$ will also fail if a weak state becomes reachable, or if the two members of a pair of weak states become reachable (these two events could be observed on the iteration if $\mathcal{S}$ did not fail).

An important point is that when queried on non-reachable chaining values, $\mathcal{S}$ uses the samplers to answer in conformance to the relation $\mathcal{R}$. However, when queried on reachable chaining values, it answers either randomly, or using the VIL-RO for consistency (but the result is still random). Thanks to this, Reach is a random subset of $\mathcal{H}$, and this allows to establish an upper-bound on the probability of failure, which directly depends on the number of weak states, and on the density of the graph representing the weak pairs of states. The view of $\mathcal{D}$ only changes if $\mathcal{S}$ aborts, and it can be shown that:

$$\mathbb{P}\big[\mathcal{S} \text{ aborts}\big] \leq 4 \cdot \frac{(q_S + 1)^2}{2^p} + |\mathcal{W}| \cdot \frac{q_S + 1}{2^p} + |\mathcal{WP}| \cdot \frac{(q_S + 1)^2}{(2^p - q_S - 1)^2}$$

Game 4: In this game, we modify the relay algorithm and leave the simulator unchanged. Instead of querying the VIL-RO, the new relay algorithm $\mathcal{T}_1$ now applies the Merkle-Damgård construction to the prefix-free encoding of its query. It uses the simulator to evaluate the compression function. Thus the relay algorithm $\mathcal{T}_1$ is essentially the same as $H^{\cdot}$, except that it is based on the simulator $\mathcal{S}$ instead of random function F.

The key argument is that the answers of $\mathcal{S}$ are consistent with those of RO: when $\mathcal{S}$ receives a sequence of queries corresponding to the prefix-free encoding of a message, it decodes it, queries the VIL-RO on the decoded message, and returns the answer of the VIL-RO. Another important detail is that before $\mathcal{S}$ fails, Reach exactly describes the reachable chaining values, and forms a tree rooted in IV. This latter property means that when a sequence of queries completes the prefix-free encoding of a message, then the message can be decoded in a unique way, which is critical in order to keep the simulator consistent with the VIL-RO.

So, all-in-all, the VIL-RO gives the same answers in Games 3 and 4, the simulator is consistent with the VIL-RO in both games, and conforms to the relation in both games. Therefore, when proceeding from Game 3 to Game 4, the view of the distinguisher only changes when $\mathcal{S}$ fails in either one of the games, but it fails more often in game 4 (because it also receives the queries of the relay algorithm).

Game 5: In this game, the VIL-RO is removed completely and the new simulator $\mathcal{S}_1$ always chooses a random p-bit response, even in situations where $\mathcal{S}$ would have consulted the VIL-RO. We also remove all the failure conditions from the new simulator $\mathcal{S}_1$.

The view of the distinguisher may only change if $\mathcal{S}$ would have failed (because now $\mathcal{S}_1$ does not).

Game 6: This is the final game of our argument. Here we finally replace the simulator $\mathcal{S}_1$ with the biased FIL-RO. Since the relay algorithm $\mathcal{T}_1$ simply implemented the prefix-free Merkle-Damgård construction, the view of the distinguisher is in fact in the construction world.

All-in-all, we find that the advantage of the distinguisher is upper-bounded by:

$$\varepsilon = 2 \cdot \mathbb{P}\big[\mathcal{S} \text{ fails in } G_3\big] + 2 \cdot \mathbb{P}\big[\mathcal{S} \text{ fails in } G_4\big]$$

And for the sake of obtaining a simpler expression, since $\mathcal{S}$ fails more often in G_4 than in G_3, we find:

$$\varepsilon \leq 4 \cdot \mathbb{P}\big[\mathcal{S} \text{ fails in } G_4\big]$$

This yields the result announced in the Theorem.

4 On Differential Attacks against SIMD

In this section we will present our results concerning differential paths in SIMD. Using Integer Linear Programming, we show that if there is a difference in the message, then the probability of the path will be at most of the order of $2^{-n/2}$. We stress that this result is not tight, but the computational power needed to improve the bound using this technique grows exponentially.

Related Work. The first attempt to avoid differential attack in a SHA/MD-like hash function was proposed in [10], where Jutla and Patthak described a linear code similar to the message expansion of SHA-1, and proved that it has a much better minimal distance than the original SHA-1 message expansion. They proposed to use SHA-1 with this new message expansion and called the new design SHA-1-IME.

Our Results. The design of SIMD follows the same idea, using a strong message expansion with a high minimal distance. In this paper we show that we can prove the security of SIMD more rigorously than the security of SHA-1-IME. While the security of SHA-1-IME is based on the heuristic assumption that the path is built out of local collisions, our proof gives an upper bound on the probability of *any* differential characteristic with a non-zero difference in the message.

Our results prove the following: for any message pair with a non-zero difference, the probability of going from an input difference Δ_{in} to an output difference Δ_{out} is bounded by 2^{-132} for SIMD-256, and 2^{-253} for SIMD-512.

4.1 Modeling Differential Paths

To study differential attacks against SIMD, we assume that the attacker builds a differential path. The differential path specifies the message difference and the state difference at each step. For each step i, we study the probability $p(i)$ that the new step difference conforms to the differential path, assuming that the previous state difference and the message difference conforms to the path, but that the values themselves are random. Since SIMD heavily uses modular additions, our analysis is based on a signed differential, as used by Wang *et al.* [18]. A signed difference gives better differential paths than an XOR difference if two active bits cancel each other out: with an XOR difference this gives a probability 1/2, but with a signed difference we have a probability 1 if the signs are opposed.

To study differential paths, we will consider the inner state of SIMD, and the Boolean functions $\phi^{(i)}$. A state bit $A_j^{(i)}$ is called *active* if it takes two different values for a message pair following the differential path. Similarly, a Boolean function is called active if at least one of its inputs is *active*. A differential path consists of a set of active message bits, active state bits, active Boolean function, and the sign of each active element. We assume that the adversary first builds such a differential path, and then looks for a conforming pair of messages and chaining values. If we disregard the first and last rounds, each Boolean function has three inputs, and each state bit enters three Boolean functions. We use this simplification in Section 4.4.

4.2 The Message Expansion

Table 3 shows the minimal distance of the message expansion of SIMD compared to the message expansion of SHA-1 and SHA-1-IME. We know that the message expansion of SIMD has a minimal distance of 520, but this is the Hamming distance, *i.e.* an XOR difference. Since we assume that the attacker will use a signed difference to build the differential path, we must study the distance of the code when the difference is given by signed binary representation. The problem is that consecutive active bits might be used as a single modular difference. For instance `0b0111` and `0b1001` differ in three bit positions, but the modular difference is only 2^1 and it can introduce a single difference in the output of a modular addition.

To compute the minimal number of modular differences introduced by the message, we use the non-adjacent form (NAF). The NAF is a signed binary representation, *i.e.* a sum of signed powers of two. It is unique and can be efficiently computed. The good property of the NAF is that it is a signed binary representation of minimal weight. For each pair of input to the inner code, we can compute the NAF of the difference, and we see that the minimal distance is 4. This means that each active word in the output of the Reed-Solomon code will introduce at least 4 differences in the state, even when we consider a differential attack using modular difference.

However, two outputs of the inner code are packed together into a 32-bit word. If we have a difference in the MSB of the low order word and in the LSB of the high order word, they can collapse to a single modular difference. In Section 4.4, we disregard this property and we just consider that the message introduces 520 differences through the message expansion. However, in Section 4.5, our model will account for that.

4.3 Structure of a Differential Path

The basic idea of our analysis is to use the lower bound on the number of active message bits to derive a lower bound on the number of active state bits. Each message difference must either introduce a new difference in the state, or cancel the propagation of a previous state difference. A single difference propagates to between 2 and 5 differences, depending on whether the Boolean

Table 3. Minimal distance of the message expansion.

	Message block	Expanded message	Minimal distance
SHA-1[1]	512 bits	1920 bits	25 bits
SHA-1-IME[1]	512 bits	1920 bits	75 bits
SIMD-256/16[2]	512 bits	2048 bits	260 bits
SIMD-512/16[2]	1024 bits	4096 bits	516 bits
SIMD-256	512 bits	4096 bits	520 bits
SIMD-512	1024 bits	8192 bits	1032 bits

[1] SHA-1 and SHA-1-IME codes are projected to the last 60 words.
[2] SIMD-$n/16$ is a reduced version using a single copy of the encoded message.

functions absorb it or let it go through. This means that a collision corresponds to between 3 and 6 message differences.

For instance, if a difference is introduced in the state $A_1^{(5)}$ by $W_1^{(5)}$, it will appear in $A_1^{(5)}$, $B_1^{(6)}$, $C_1^{(7)}$, $D_1^{(8)}$. Each of the Boolean function $\phi_1^{(6)}, \phi_1^{(7)}, \phi_1^{(8)}$ can either absorb it or pass it. This difference will propagate to $A_0^{(6)}$, and to $A_1^{(9)}$. Moreover, it can propagate to $A_1^{(6)}$, $A_1^{(7)}$ and $A_1^{(8)}$ if the Boolean functions do not absorb it. Up to five active message bits can be used to cancel this propagation: $W_1^{(4)}$, $W_1^{(8)}$, $W_0^{(5)}$, and possibly $W_1^{(5)}$, $W_1^{(6)}$, $W_1^{(7)}$ if the corresponding Boolean functions are not absorbing.

We consider two parts of the compression function: the computation of ϕ, and the modular sum. In order to study the probabilities associated with these computations, we will count the conditions needed for a message pair to follow the characteristic.

ϕ-conditions. The Boolean functions MAJ and IF used in SIMD can either absorb or pass differences. When there is a single active input, the probability to absorb and to pass is $1/2$. Each time a state difference enters a Boolean function, the differential characteristic specifies whether the difference should be passed or absorbed, and this gives one condition if the Boolean functions have a single active input. Thus, each isolated difference in the state will account for 3 ϕ-conditions: one for each Boolean function they enter. For instance, a difference in $A_1^{(4)}$ generates conditions for $\phi_1^{(6)}$, $\phi_1^{(7)}$, $\phi_1^{(8)}$.

$\boxplus$-conditions. When a difference is introduced in the state, it has to come from one of the inputs of the round function:

$$A_j^{(i)} = \left(D_j^{(i-1)} \boxplus W_j^{(i)} \boxplus \phi^{(i)}(A_j^{(i-1)}, B_j^{(i-1)}, C_j^{(i-1)})\right)^{\lll s^{(i)}} \boxplus \left(A_{p^{(i)}(j)}^{(i-1)}\right)^{\lll r^{(i)}}$$

The round function is essentially a sum of 4 terms, and the differential characteristic will specify which input bits and which output bits are active. Thus, the differential characteristic specifies how the carry should propagate, and this gives at least one condition per state difference.

In the end, a state difference accounts for 4 conditions.

4.4 Heuristics

We first give some results based on heuristics. We assume that the adversary can find message pairs that give a minimal distance in the expanded message, and we allow him to add some more constraints to the expanded message. Note that finding a message pair with a low difference in the expanded message is already quite difficult with the message expansion of SIMD.

Heuristic I assumes that the adversary can find message pairs with minimal distance, but no other useful property. The adversary gets a message pair with minimal distance, and connects the dots to build a differential characteristic.

Heuristic II assumes that the adversary can find message pairs with minimal distance and controls the relative positions of the message difference. He will use that ability to create local collisions.

Heuristic III assumes that the adversary can find a message pair with any message difference, limited only by the minimal weight of the code. He will cluster local collisions to avoid many conditions.

Heuristic I. In this section, we assume that the adversary can find a message pair such that the expanded messages reach the minimal distance of the code, but we assume that the message pair has no further useful properties.

In this case, this adversary gets a message pair with a small difference and he has to connect the dots to build a differential path. This is somewhat similar to the attacks on MD4 [17]: the messages are chosen so as to make a local collision in the last round, and the attacker has to connect all the remaining differences into a path with a good probability.

It seems safe to assume that such a differential path will at least have as many active state bits as active message bits. Since an isolated difference in the state costs 4 conditions, we expect at least 2080 conditions (resp. 4128 for SIMD-512), which is very high.

This shows that the adversary needs some control over the expanded message. If he wants to succeed, he needs to find message pairs with some extra properties.

Heuristic II. We now assume that the adversary can force some structure in the expanded message difference. Namely, he can choose the relative location of the differences in the expanded message. Since the probability of the path is essentially given by the number of active bits in the state, the path should minimize this. This is achieved with local collisions, and each local collision will use as many message differences as possible. Due to the structure of the round function of SIMD, a local collision can use between 3 and 6 message differences, depending on whether the Boolean functions absorb or pass the differences. In order to minimize the number of state differences, the path will make all the Boolean functions pass the differences, yielding six message differences per state difference. This is somewhat counter-intuitive because most attacks try to minimize the propagation of differences by absorbing them. However, in our case it is more efficient to let the differences go through the Boolean functions, and to use more message differences to cancel them, because we have a lower bound on the number of message differences.

Since the adversary only controls the relative position of the message differences, we assume that most local collisions will be isolated, so that each local collision gives 4 conditions. Thus, a differential is expected to have at least $520 \times 4/6 \approx 347$ conditions (688 for SIMD-512). This leaves a significant security margin, and even if the adversary can use message modifications in the first 16 rounds, it can only avoid half of those conditions.

This can be compared to the attacks on SHA-1 [6,18]. These attacks are based on local collisions, but we do not know how to find a message pair which would have both minimal distance and yield a series of local collisions in SHA-1. Instead, attacks on SHA-1 use the fact that the message expansion is *linear* and *circulant*: given a codeword, if we shift it by a few rounds we get another valid codeword and similarly if we rotate each word we get another valid codeword. Then we can combine a few rotated and/or shifted codewords so as to build local collisions. The

Program 1 Linear Program

Minimize $S + \alpha - \beta$ **with the constraints:**

$$3S = \alpha + \beta + \gamma \tag{1}$$
$$520 \leq 3S + \alpha \tag{2}$$
$$\gamma \leq \beta \leq \alpha \tag{3}$$

$\alpha \geq 0$	is the number of Boolean functions with at least one active input
$\beta \geq 0$	is the number of Boolean functions with at least two active inputs
$\gamma \geq 0$	is the number of Boolean functions with at least three active inputs
$S \geq 0$	is the number of active state bits

attacks on SHA-1 start with a codeword of minimal distance, and combines 6 rotated versions. Thus the weight of the actual expanded message difference used in the attack is six times the minimal weight of the code.

Note that message expansion of SIMD is more complex than the one from SHA-1, and it seems very hard to find this kind of message pairs in SIMD. Moreover, the trick used in SHA-1 cannot be used here because the message expansion is neither linear nor circulant.

Heuristic III. We now remove all heuristic assumptions and we try to give a bound on *any* differential trail. However, to keep this analysis simple, we still disregard the specificities of the first round, and the fact that one can combine some of the message differences.

The adversary will still use local collisions to minimize the number of differences in the state, but he will also try to reduce the number of conditions for each local collision by clustering them. We have seen that an isolated state difference costs 4 conditions, but if two state differences are next to each other, the cost can be reduced when using a signed difference. For instance, if two inputs of the MAJ function are active, the adversary does not have to pay any probability: if both active inputs have the same sign, then the output is active with the same sign, but if the inputs have opposite signs then the output will be inactive. In this section we consider that a Boolean function with more than one active input does not cost any probability.

Thus, the best strategy for the adversary is to place the state differences so that each active Boolean function has two active inputs, in order to avoid any ϕ-conditions. Each state difference costs only one $\boxplus$-condition, and gets 4.5 message differences (these message differences corresponding to the Boolean functions are shared between two Boolean functions). This gives a lower bound of 116 conditions.

More rigorously, this can be described by a linear program, as shown in Linear Program 1. Equation (1) comes from counting the number of active inputs to the Boolean functions in two different ways, while Equation (2) counts the number of message differences that can be used. The objective value $S + \alpha - \beta$ counts the conditions: one for each state difference, plus one for each Boolean function with exactly one active input. The optimal solution to this program is $520/4.5 \approx 115.55$.

In the next section we will see how to improve this bound and get a bound on the probability of any differential path.

Comparison with SHA-1-IME. The security of SHA-1-IME is based on a heuristic that is quite similar to our Heuristic I. Jutla and Patthak assume that the adversary will use the same technique as the attacks on SHA-1, *i.e.* create local collisions using the fact that the code is linear and circulant. They deduce that the probability of a differential characteristic will be about

$2^{75\times 2.5}$. They implicitly assume that the adversary cannot find minimal codewords that would already give local collisions. Our Heuristic II assumes that the attacker can find such codewords, and if we apply it to SHA-1-IME, it would only guarantee that we have at least 13 local collisions (each local collision accounts for 6 message differences). Since a local collision in SHA-1 has an average probability of $2^{-2.5}$, this would only prove that an attack has at least a complexity $2^{13\times 2.5} = 2^{32.5}$.

This shows that our Heuristic II and III are much weaker than the heuristic used in SHA-1-IME.

4.5 Upper Bounding the Probability of a Differential Path

The bound given by Heuristic III is slightly lower than $n/2$ so we would like to improve it. To find a better bound, we will follow the approach of Linear Program 1. Note that in the optimal solution, all the Boolean functions have either zero or two active inputs, but it is unlikely that such a path actually exists because of the way the Boolean functions share inputs. In order to remove some impossible solutions, we use a more detailed modeling of differential paths where each individual state bit is treated separately. This also allows us to express some extra constraints that will help to improve the lower bound.

Program 2 Integer Linear Program (simplified)

Minimize $\sum S_i^{(j)[k]} + \sum \alpha_i^{(j)[k]} - \sum \beta_i^{(j)[k]}$ **with the constraints:**

$$S_i^{(j-1)[k]} + S_i^{(j-2)[k]} + S_i^{(j-3)[k]} = \alpha_i^{(j)[k]} + \beta_i^{(j)[k]} + \gamma_i^{(j)[k]} \tag{1'}$$

$$W_i^{(j)[k]} \leq S_i^{(j)[k+s_j]} + S_i^{(j-4)[k-r_j]} + S_{p_j(i)}^{(j-1)[k-r_j+s_j]} + \alpha_i^{(j)[k]} \tag{2'}$$

$$\gamma_i^{(j)[k]} \leq \beta_i^{(j)[k]} \leq \alpha_i^{(j)[k]} \tag{3'}$$

$$\sum_{k=0}^{15} W_i^{(j)[k]} \geq 4Y_{P_1(i,j)} \qquad \sum_{k=16}^{31} W_i^{(j)[k]} \geq 4Y_{P_0(i,j)} \tag{4}$$

$$\sum Y_i \geq 65 \tag{5}$$

$\alpha_i^{(j)[k]} \in \mathcal{B}$	is true iff $\phi_i^{(j)[k]}$ has at least one active input
$\beta_i^{(j)[k]} \in \mathcal{B}$	is true iff $\phi_i^{(j)[k]}$ has at least two active input
$\gamma_i^{(j)[k]} \in \mathcal{B}$	is true iff $\phi_i^{(j)[k]}$ has at least three active input
$S_i^{(j)[k]} \in \mathcal{B}$	is true iff the state bit $A_i^{(j)[k]}$ is active
$W_i^{(j)[k]} \in \mathcal{B}$	is true iff the expended message bit $W_i^{(j)[k]}$ is active
$Y_i \in \mathcal{B}$	is true iff the word i is active in the output of the NTT

Constraints related to the message expansion. We know that the message expansion gives at least 520 differences in the expanded message, but there are some constraints on the positions of these differences. Namely, we have at least 65 active words in each copy of the message, and each active word has at least 4 active bits. For instance, a difference pattern with 3 active bits in each word would have 768 bit differences, but it is not a valid pattern. Moreover, the active words in both copies have to be the same up to the permutation P. To include these constraints in our model, we add a set of binary variables Y_i which encode whether word i is active in the output of the NTT. This is modeled by Equations (4) and (5). Note that this still allows many difference patterns that cannot be the output of a real message pair.

Better cost estimation. In Program 1, we only count a condition for the Boolean functions with a single active input. In fact, if we look at the truth table of the Boolean functions we see that the IF function still needs a condition when inputs 1 and 2, or 1 and 3 are active. Since we are using distinct variables for each of these inputs, we can include this in our description.

We can write all these constraints as a huge optimisation problem, but we need some tool to find the optimal solution of the system, or at least find a lower bound. We decided to write our problem as an Integer Linear Program.

Integer Linear Programming. Integer Linear Programming (ILP) is a generalisation of Linear Programming (LP) where some variables are restricted to integer values. While LP is solvable in polynomial time, ILP is NP-complete. ILP solvers usually use some variants of the branch-and-bound algorithm. In the case of minimization problem, the branch-and-bound algorithm computes a lower bound to the optimal solution and incrementally raises this lower bound. Meanwhile, non-optimal solutions give an upper bound, and when the two bounds meet, the search is over.

To compute a lower bound, the problem is relaxed by considering all variables as real numbers instead of integers. This gives a Linear Program which can be solved efficiently, and the optimal solution of the Linear Program is a lower bound of the Integer Linear Program. To improve this lower bound, the search space is divided into two or more subspaces, and a lower bound is computed recursively for each subproblem. For instance, Figure 4 shows how to solve Program 1 as an ILP.

Results. A simplified version of the ILP is given by Program 2. The first equations and the objective value mirrors Program 1, but use many variables to allow for more precise extra constraints. The full program has 28,576 variables and 80,162 equations for SIMD-256. We used the solver SYMPHONY, an open-source solver for mixed-integer linear programs, available at `http://www.coin-or.org/SYMPHONY/`. The solver could not find an optimal solution to the program, but it reached an interesting lower bound after some time: a differential path for SIMD-256 has at least 132 conditions, while a differential path for SIMD-512 has at least 253. The computation for SIMD-512 took one month on a bi-quadcore machine.

Summary. The optimal strategy of the attacker is to use local collisions (avoiding any difference propagation) and to cluster the local collisions so as to avoid most conditions. Our modeling allows the adversary to do this because he can choose the message difference and the expanded message difference independently, and he can position the differences arbitrarily in the inner code. However, this is not possible in practice, and most solutions of the Integer Linear Program will require an expanded message difference that is not actually feasible. It should also be noted that we do not model the sign of the differences, and we always assume that the sign is correct when two differences cancel out.

Therefore, we expect that the best differential path in SIMD is much worse that the optimal solution of our Integer Linear Program. Moreover, the program is too large to be solved to optimality, and we only have a lower bound on the number of conditions (this lower bound keep improving if we let the solver run).

Limitations

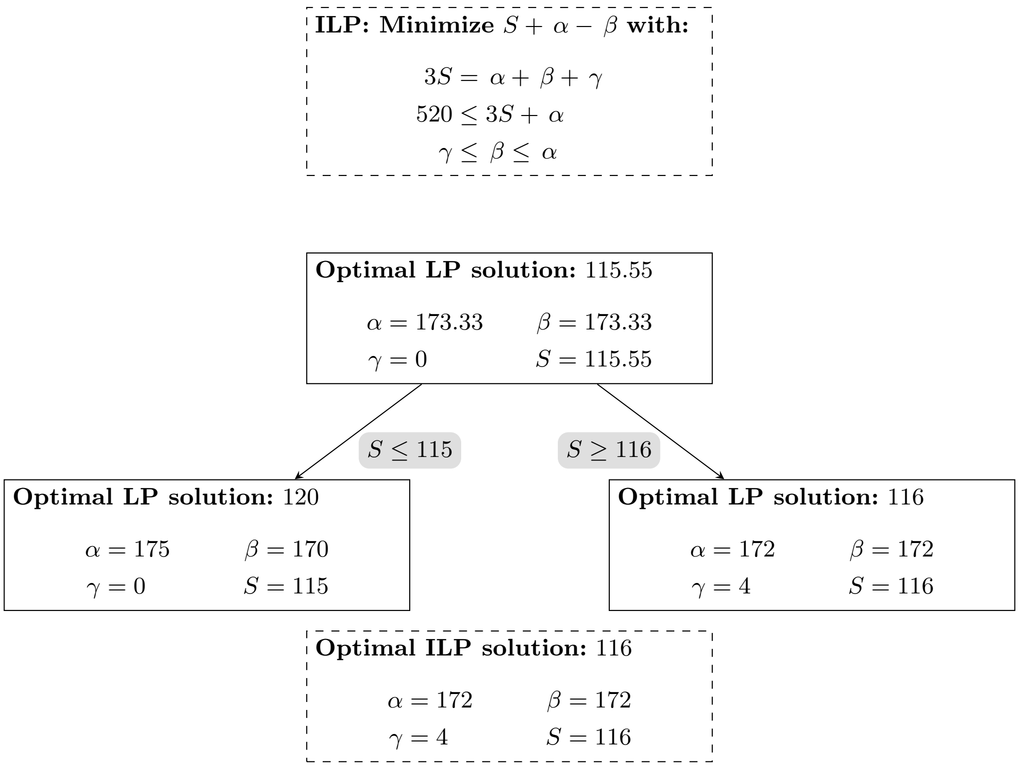
ILP: Minimize $S + \alpha - \beta$ **with:**
$3S = \alpha + \beta + \gamma$
$520 \leq 3S + \alpha$
$\gamma \leq \beta \leq \alpha$
Optimal LP solution: 115.55
$\alpha = 173.33$ $\beta = 173.33$
$\gamma = 0$ $S = 115.55$
$S \leq 115$
$S \geq 116$
Optimal LP solution: 120
$\alpha = 175$ $\beta = 170$
$\gamma = 0$ $S = 115$
Optimal LP solution: 116
$\alpha = 172$ $\beta = 172$
$\gamma = 4$ $S = 116$
Optimal ILP solution: 116
$\alpha = 172$ $\beta = 172$
$\gamma = 4$ $S = 116$

Fig. 4. Solving Program 1 as a ILP. The problem is divided into two subproblems: $S \leq 115$ ans $S \geq 116$. For each of these subproblems, the optimal solution of the relaxed problem is integral so this gives us the optimal solution of the ILP.

About Message Modifications When we consider the Heuristic III, our proof does not leave enough margin to account for message modifications. However the mode of operation of the compression function is designed to make message modification difficult, by XORing the message and the chaining value in the very beginning. This prevents usual message modification techniques, because the adversary has to commit to some message before he can begin the real computation of the compression function.

We note that given a message M, one can compute a new message M^* so that the expanded message W^* is identical to W in the first steps, up to almost 8 steps. However in order to keep the same state in the Feistel rounds, one has to counter the modification of the message by a modification of the input chaining value. Therefore it is only applicable to free-start attacks. Since SIMD is a wide-pipe design, free-start attacks on the compression function cannot be turned into attacks on the hash function.

Redundant Conditions There might be some redundant ϕ-conditions in a differential path. As opposed to MD4 or MD5, we can never have the same condition for two different Boolean functions (because of the rotations), but there might still be some redundant conditions. However, since most of the conditions are $\boxplus$-conditions, we believe this is negligible.

5 Security Status of SIMD

5.1 On the Symmetry-based Distinguisher

The distinguisher of Section 2 shows that the compression function of SIMD is not ideal. It does not affect the security of the hash function, but it is nonetheless an unwanted property. Since this distinguisher is based on symmetry properties, it is easy to avoid this property by slightly changing the design. Therefore, we plan to tweak the SIMD design by adding non-symmetric constants, if given such an opportunity. We also note that other SHA-3 candidates are in a similar situation:

- CubeHash has strong symmetry properties in its round transformation [1,8]. It is thought that since the initial state in not symmetric, it is not possible to reach a symmetric state.
- Shabal has strong distinguishers on its compression function: there are differential paths with probability 1 [2], and the inverse permutation does not have full diffusion (some input bits do not depend on all output bits). The Shabal team has shown that these distinguishers do not affect the security [5].

Countermeasures. An interesting way to avoid the symmetry properties would be to add a counter to the expanded message after the multiplication by a constant (step 3 of the message expansion). This would ensure that each expanded message word has a different value modulo 185 (respectively modulo 223), and it prevents equality constraints between the expanded message words.

5.2 On Differential Attacks

Concerning differential attacks, our results are two-fold:

1. A differential path with a non-zero difference in the input chaining value does not affect the security of the hash function because it is wide-pipe
2. A differential path with a non-zero difference in the message cannot have a high success probability, because of the strong message expansion.

This shows that successful attacks on the hash function based on differential properties are very unlikely.

Acknowledgments

We would like to thank Praveen Gauravaram from Technical University of Denmark, Copenhagen for discussions on the proof of indifferentiability.

We would also like to thank Franck Landelle from CELLAR for insightful comments on the security of SIMD and limitations of our initial study of differential paths. Part of this work was supported by CELLAR.

Part of this work was supported by the European Commission through ECRYPT, and by the French government through the Saphir RNRT project.

References

1. Aumasson, J.P., Brier, E., Meier, W., Naya-Plasencia, M., Peyrin, T.: Inside the Hypercube. In Boyd, C., Nieto, J.M.G., eds.: ACISP. Volume 5594 of Lecture Notes in Computer Science., Springer (2009) 202–213
2. Aumasson, J.P., Mashatan, A., Meier, W.: More on Shabal's permutation. OFFICIAL COMMENT (2009)
3. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: ASIACRYPT. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 1–18
4. Bouillaguet, C., Dunkelman, O., Fouque, P.A., Leurent, G.: Another Look at the Complementation Property. In Hong, S., Iwata, T., eds.: FSE '10. Lecture Notes in Computer Science, Springer (2010)
5. Bresson, E., Canteaut, A., Chevallier-Mames, B., Clavier, C., Fuhr, T., Gouget, A., Icart, T., Misarsky, J.F., Naya-Plasencia, M., Paillier, P., Pornin, T., Reinhard, J.R., Thuillet, C., Videau, M.: Indifferentiability with Distinguishers: Why Shabal Does Not Require Ideal Ciphers. Cryptology ePrint Archive, Report 2009/199 (2009) `http://eprint.iacr.org/`.
6. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In Krawczyk, H., ed.: CRYPTO. Volume 1462 of Lecture Notes in Computer Science., Springer (1998) 56–71
7. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: CRYPTO'05. (2005) 430–448
8. Ferguson, N., Lucks, S., McKay, K.A.: Symmetric States and their Structure: Improved Analysis of CubeHash. Cryptology ePrint Archive, Report 2010/273 (2010) `http://eprint.iacr.org/`.
9. Gauravaram, P., Bagheri, N. Private communication (July 2009)
10. Jutla, C.S., Patthak, A.C.: Provably Good Codes for Hash Function Design. In Biham, E., Youssef, A.M., eds.: Selected Areas in Cryptography. Volume 4356 of Lecture Notes in Computer Science., Springer (2006) 376–393
11. Leurent, G., Bouillaguet, C., Fouque, P.A.: SIMD Is a Message Digest. Submission to NIST (2008)
12. Leurent, G., Bouillaguet, C., Fouque, P.A.: Tweaking SIMD. Submission to the second round of SHA-3 competition (2009)
13. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Naor, M., ed.: TCC. Volume 2951 of Lecture Notes in Computer Science., Springer (2004) 21–39
14. Mendel, F., Nad, T.: A Distinguisher for the Compression Function of SIMD-512. In Roy, B.K., Sendrier, N., eds.: INDOCRYPT. Volume 5922 of Lecture Notes in Computer Science., Springer (2009) 219–232
15. Nikolić, I., Pieprzyk, J., Sokolowski, P., Steinfeld, R.: Rotational Cryptanalysis of (Modified) Versions of BMW and SIMD. NIST hash forum (March 2010)
16. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In Johansson, T., ed.: FSE. Volume 2887 of Lecture Notes in Computer Science., Springer (2003) 247–260
17. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 1–18
18. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In Shoup, V., ed.: CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 17–36
19. Yu, H., Wang, X.: Cryptanalysis of the compression function of simd. Cryptology ePrint Archive, Report 2010/304 (2010) `http://eprint.iacr.org/`.

A Example of Weak Message

Here is an example of a weak message, and the output of the compression function when used with the same value for the message and chaining value (this ensures that the XOR is symmetric). Notice that the output is mostly symmetric.

Message = Chaining Value					Output				
$A_{0..3}$	00000000	00000000	00000000	00000000	$A_{0..3}$	0e0618e6	0ee618e6	ec5a3cee	fbdc48ae
$B_{0..3}$	00000000	00000000	00000000	a2000000	$B_{0..3}$	17bde794	17bddbd4	5a0a59f2	5a2a59f2
$C_{0..3}$	00000000	00000000	00000000	00000000	$C_{0..3}$	12a9c015	12a9c015	be7d3df1	be775df1
$D_{0..3}$	00000000	00000000	00000000	f1000000	$D_{0..3}$	15f9cb8d	15f9cb8d	2efef45c	2efef45c

Expanded Message									
$W^{(0)}_{0..3}$	0a1ee3d1	0a1ee3d1	bc12531b	bc12531b	$W^{(1)}_{0..3}$	a5abca86	a5abca86	4be14335	4be14335
$W^{(2)}_{0..3}$	e827b082	e827b082	1b761da1	1b761da1	$W^{(3)}_{0..3}$	287848fd	287848fd	aa01d8fa	aa01d8fa
$W^{(4)}_{0..3}$	2fb2e543	2fb2e543	c914c4be	c914c4be	$W^{(5)}_{0..3}$	050f4ec5	050f4ec5	de09ccb1	de09ccb1
$W^{(6)}_{0..3}$	143cc7a2	143cc7a2	31ddec7d	31ddec7d	$W^{(7)}_{0..3}$	50f0d841	50f0d841	0dbbb1f4	0dbbb1f4
$W^{(8)}_{0..3}$	d04e1abd	d04e1abd	36ec3b42	36ec3b42	$W^{(9)}_{0..3}$	ebc4385e	ebc4385e	ce231383	ce231383
$W^{(10)}_{0..3}$	f5e21c2f	f5e21c2f	43eeace5	43eeace5	$W^{(11)}_{0..3}$	17d94f7e	17d94f7e	e48ae25f	e48ae25f
$W^{(12)}_{0..3}$	af1027bf	af1027bf	f2454e0c	f2454e0c	$W^{(13)}_{0..3}$	faf1b13b	faf1b13b	21f7334f	21f7334f
$W^{(14)}_{0..3}$	d788b703	d788b703	55ff2706	55ff2706	$W^{(15)}_{0..3}$	5a55357a	5a55357a	b41fbccb	b41fbccb
$W^{(16)}_{0..3}$	320fcdf1	320fcdf1	624c9db4	624c9db4	$W^{(17)}_{0..3}$	a4135bed	a4135bed	3126ceda	3126ceda
$W^{(18)}_{0..3}$	21adde53	21adde53	4aa2b55e	4aa2b55e	$W^{(19)}_{0..3}$	237fdc81	237fdc81	975568ab	975568ab
$W^{(20)}_{0..3}$	435abca6	435abca6	ab5b54a5	ab5b54a5	$W^{(21)}_{0..3}$	9ccb6335	9ccb6335	409fbf61	409fbf61
$W^{(22)}_{0..3}$	641e9be2	641e9be2	daaf2551	daaf2551	$W^{(23)}_{0..3}$	46feb902	46feb902	1893e76d	1893e76d
$W^{(24)}_{0..3}$	71c58e3b	71c58e3b	a06f5f91	a06f5f91	$W^{(25)}_{0..3}$	1e09e1f7	1e09e1f7	dd6a2296	dd6a2296
$W^{(26)}_{0..3}$	9a1065f0	9a1065f0	eeb5114b	eeb5114b	$W^{(27)}_{0..3}$	c3ee3c12	c3ee3c12	452cbad4	452cbad4
$W^{(28)}_{0..3}$	e684197c	e684197c	c1333ecd	c1333ecd	$W^{(29)}_{0..3}$	f9a1065f	f9a1065f	2ac7d539	2ac7d539
$W^{(30)}_{0..3}$	f3420cbe	f3420cbe	558eaa72	558eaa72	$W^{(31)}_{0..3}$	cd0832f8	cd0832f8	6c4f93b1	6c4f93b1

B Study of the Symmetry Classes

In this section we study the sets of messages than be used for the symmetry property of SIMD by solving the equation on the NTT. We give an explicit description of the sets, and we show that for a given symmetry relation, a pair of output cannot be used as input for the symmetry property.

B.1 Symmetries for SIMD-256

There are three symmetry classes in SIMD-256.

Class 1: $\overset{\leftrightarrow\leftrightarrow}{\bullet}$ **($y_i = y'_{i\oplus 2}$, $W_i = W'_{i\oplus 1}$).** For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$M_{\alpha,\beta} = 162 \times e_{31} + 241 \times e_{63} + \alpha \times e_0 + \beta \times e_{32}$$
$$M'_{\alpha,\beta} = 162 \times e_{31} + 241 \times e_{63} + \alpha \times e_0 - \beta \times e_{32}$$

Note that if $\beta = 1$ we have $-\beta = -1$ and the corresponding M' is not a valid message. So we have only $256 \cdot 255$ valid message pairs.

When $\beta = 0$, we have $M_{\alpha,\beta} = M_{\alpha,\beta}$ and this gives a symmetric message instead of a pair of symmetric message.

Class 2: $\underset{\leftrightarrow}{\bullet}$ $(y_i = y'_{i\oplus 4},\ W_i = W'_{i\oplus 2})$. For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 4:

$$\begin{aligned} M_{\alpha,\beta,\gamma,\delta} &= 55 \times e_{15} + 232 \times e_{31} + 37 \times e_{47} + 16 \times e_{63} + \alpha \times e_0 + \beta \times e_{32} + \gamma \times e_{16} + \delta \times e_{48} \\ M'_{\alpha,\beta,\gamma,\delta} &= 55 \times e_{15} + 232 \times e_{31} + 37 \times e_{47} + 16 \times e_{63} + \alpha \times e_0 + \beta \times e_{32} - \gamma \times e_{16} - \delta \times e_{48} \end{aligned}$$

Class 3: $\overset{\leftrightarrow}{\underset{\leftrightarrow}{\bullet}}$ $(y_i = y'_{i\oplus 6},\ W_i = W'_{i\oplus 3})$. For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$\begin{aligned} M_{\alpha,\beta} &= 212 \times e_{15} + 181 \times e_{31} + 139 \times e_{47} + 20 \times e_{63} + \alpha \times e_0 + \beta \times e_{32} \\ M'_{\alpha,\beta} &= 212 \times e_{15} + 181 \times e_{31} + 139 \times e_{47} + 20 \times e_{63} + \alpha \times e_0 - \beta \times e_{32} \end{aligned}$$

Input and Output Pairs. Let M, M' be a message in one of these classes. Without loss of generality, we denote the symmetry class by $\overleftrightarrow{\bullet}$.

Let h, h' be a pair of chaining values that can used with this message, *i.e.*, $h' = \overleftrightarrow{h} \oplus M \oplus M'$. We denote the inputs to the Feistel compression part by $\mathcal{S}^{(0)} = A^{(0)}_{[0,1,2,3]}, B^{(0)}_{[0,1,2,3]}, C^{(0)}_{[0,1,2,3]}, D^{(0)}_{[0,1,2,3]}$ (respectively $\mathcal{S}'^{(0)}$). Similarly, $\mathcal{S}^{(31)}$ is the state after the 32 Feistel rounds using the message, and $\mathcal{S}^{(35)}$ if the final state after the Feed-forward.

We can express $D^{(35)}_3$ in terms of $\mathcal{S}^{(31)}$:

$$\begin{aligned} D'^{(35)}_3 &= \left(\left(D'^{(31)}_3 \boxplus IV'_3 \boxplus \mathsf{IF}(A'^{(31)}_3, B'^{(31)}_3, C'^{(31)}_3) \right)^{\lll 13} \boxplus A'^{(31)\lll 4}_0 \right)^{\lll 13} \\ \overleftrightarrow{D}^{(35)}_3 &= \left(\left(\overleftrightarrow{D}^{(31)}_3 \boxplus \overleftrightarrow{IV}_3 \boxplus \mathsf{IF}(\overleftrightarrow{A}^{(31)}_3, \overleftrightarrow{B}^{(31)}_3, \overleftrightarrow{C}^{(31)}_3) \right)^{\lll 13} \boxplus \overleftrightarrow{A}^{(31)\lll 4}_0 \right)^{\lll 13} \end{aligned}$$

Since $\mathcal{S}^{(31)}$ and $\mathcal{S}'^{(31)}$ are symmetric, we have:

$$A'^{(31)}_3 = \overleftrightarrow{A}^{31}_3 \qquad B'^{(31)}_3 = \overleftrightarrow{B}^{31}_3 \qquad C'^{(31)}_3 = \overleftrightarrow{C}^{31}_3 \qquad D'^{(31)}_3 = \overleftrightarrow{D}^{31}_3 \qquad A'^{(31)}_0 = \overleftrightarrow{A}^{31}_0$$

Therefore, the difference between $D'^{(35)}_3$ and $\overleftrightarrow{D}^{(35)}_3$ comes from the difference between IV'_3 and $\overleftrightarrow{IV}_3$. By looking at those differences, we can see that they are not compatible, because of the rotation by 26 bits:

Class 1	$IV'_3 \oplus \overleftrightarrow{IV}_3 = 0$,	$D'^{(35)}_3 \oplus \overleftrightarrow{D}^{(35)}_3 = 241 \times 2^{24}$
Class 2	$IV'_3 \oplus \overleftrightarrow{IV}_3 = 55 \times 2^{24}$,	$D'^{(35)}_3 \oplus \overleftrightarrow{D}^{(35)}_3 = 16 \times 2^{24}$
Class 3	$IV'_3 \oplus \overleftrightarrow{IV}_3 = 212 \times 2^{24}$,	$D'^{(35)}_3 \oplus \overleftrightarrow{D}^{(35)}_3 = 20 \times 2^{24}$

B.2 Symmetries for SIMD-512

There are seven symmetry classes for SIMD-512.

Class 1 $(y_i = y'_{i\oplus 2},\ W_i = W'_{i\oplus 1})$. For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$\begin{aligned} M_{\alpha,\beta} &= 180 \times e_{63} + 241 \times e_{127} + \alpha \times e_0 + \beta \times e_{64} \\ M'_{\alpha,\beta} &= 180 \times e_{63} + 241 \times e_{127} + \alpha \times e_0 - \beta \times e_{64} \end{aligned}$$

Class 2 ($y_i = y'_{i\oplus 4}$, $W_i = W'_{i\oplus 2}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 4:

$$\begin{aligned}
M_{\alpha,\beta,\gamma,\delta} &= 74 \times e_{31} + 232 \times e_{63} + 218 \times e_{95} + 16 \times e_{127} + \alpha \times e_0 + \beta \times e_{64} + \gamma \times e_{32} + \delta \times e_{96} \\
M'_{\alpha,\beta,\gamma,\delta} &= 74 \times e_{31} + 232 \times e_{63} + 218 \times e_{95} + 16 \times e_{127} + \alpha \times e_0 + \beta \times e_{64} - \gamma \times e_{32} - \delta \times e_{96}
\end{aligned}$$

Class 3 ($y_i = y'_{i\oplus 6}$, $W_i = W'_{i\oplus 3}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$\begin{aligned}
M_{\alpha,\beta} &= 58 \times e_{31} + 16 \times e_{63} + 150 \times e_{95} + 122 \times e_{127} + \alpha \times e_0 + \beta \times e_{64} \\
M'_{\alpha,\beta} &= 58 \times e_{31} + 16 \times e_{63} + 150 \times e_{95} + 122 \times e_{127} + \alpha \times e_0 - \beta \times e_{64}
\end{aligned}$$

Class 4 ($y_i = y'_{i\oplus 8}$, $W_i = W'_{i\oplus 4}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 8:

$$\begin{aligned}
M_{\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\eta,\theta} &= 211 \times e_{15} + 8 \times e_{31} + 199 \times e_{47} + 234 \times e_{63} + 116 \times e_{79} + 32 \times e_{95} + 111 \times e_{111} + 16 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} + \gamma \times e_{32} + \delta \times e_{96} + \varepsilon \times e_{16} + \zeta \times e_{48} + \eta \times e_{80} + \theta \times e_{112} \\
M'_{\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\eta,\theta} &= 211 \times e_{15} + 8 \times e_{31} + 199 \times e_{47} + 234 \times e_{63} + 116 \times e_{79} + 32 \times e_{95} + 111 \times e_{111} + 16 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} + \gamma \times e_{32} + \delta \times e_{96} - \varepsilon \times e_{16} - \zeta \times e_{48} - \eta \times e_{80} - \theta \times e_{112}
\end{aligned}$$

Class 5 ($y_i = y'_{i\oplus 10}$, $W_i = W'_{i\oplus 5}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$\begin{aligned}
M_{\alpha,\beta} &= 195 \times e_{15} + 237 \times e_{31} + 154 \times e_{47} + 254 \times e_{63} + 70 \times e_{79} + 40 \times e_{95} + 121 \times e_{111} + 195 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} \\
M'_{\alpha,\beta} &= 195 \times e_{15} + 237 \times e_{31} + 154 \times e_{47} + 254 \times e_{63} + 70 \times e_{79} + 40 \times e_{95} + 121 \times e_{111} + 195 \times e_{127} \\
&\quad + \alpha \times e_0 - \beta \times e_{64}
\end{aligned}$$

Class 6 ($y_i = y'_{i\oplus 12}$, $W_i = W'_{i\oplus 6}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 4:

$$\begin{aligned}
M_{\alpha,\beta,\gamma,\delta} &= 251 \times e_{15} + 35 \times e_{31} + 36 \times e_{47} + 223 \times e_{63} + 57 \times e_{79} + 159 \times e_{95} + 0 \times e_{111} + 114 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} + \gamma \times e_{32} + \delta \times e_{96} \\
M'_{\alpha,\beta,\gamma,\delta} &= 251 \times e_{15} + 35 \times e_{31} + 36 \times e_{47} + 223 \times e_{63} + 57 \times e_{79} + 159 \times e_{95} + 0 \times e_{111} + 114 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} - \gamma \times e_{32} - \delta \times e_{96}
\end{aligned}$$

Class 7 ($y_i = y'_{i\oplus 14}$, $W_i = W'_{i\oplus 7}$). For this symmetry relation, the pairs of suitable messages are in an affine space of dimension 2:

$$\begin{aligned}
M_{\alpha,\beta} &= 32 \times e_{15} + 212 \times e_{31} + 157 \times e_{47} + 218 \times e_{63} + 129 \times e_{79} + 162 \times e_{95} + 174 \times e_{111} + 199 \times e_{127} \\
&\quad + \alpha \times e_0 + \beta \times e_{64} \\
M'_{\alpha,\beta} &= 32 \times e_{15} + 212 \times e_{31} + 157 \times e_{47} + 218 \times e_{63} + 129 \times e_{79} + 162 \times e_{95} + 174 \times e_{111} + 199 \times e_{127} \\
&\quad + \alpha \times e_0 - \beta \times e_{64}
\end{aligned}$$

Input and Output Pairs. Like in SIMD-256, we can show than a pair of output cannot be a valid input pair for the symmetry relation. We have:

$$D'^{(35)}_7 = \left(\left(D'^{(31)}_7 \boxplus IV'_7 \boxplus \mathsf{IF}(A'^{(31)}_7, B'^{(31)}_7, C'^{(31)}_7)\right)^{\lll 13} \boxplus A'^{(31)\lll 4}_2\right)^{\lll 13}$$

$$\overleftrightarrow{D}^{(35)}_7 = \left(\left(\overleftrightarrow{D}^{(31)}_7 \boxplus \overleftrightarrow{IV}_7 \boxplus \mathsf{IF}(\overleftrightarrow{A}^{(31)}_7, \overleftrightarrow{B}^{(31)}_7, \overleftrightarrow{C}^{(31)}_7)\right)^{\lll 13} \boxplus \overleftrightarrow{A}^{(31)\lll 4}_2\right)^{\lll 13}$$

Since $\mathcal{S}^{(31)}$ and $\mathcal{S}'^{(31)}$ are symmetric, we have:

$$A'^{(31)}_7 = \overleftrightarrow{A}^{31}_7 \qquad B'^{(31)}_7 = \overleftrightarrow{B}^{31}_7 \qquad C'^{(31)}_7 = \overleftrightarrow{C}^{31}_7 \qquad D'^{(31)}_7 = \overleftrightarrow{D}^{31}_7 \qquad A'^{(31)}_2 = \overleftrightarrow{A}^{31}_2$$

Again, the differences in $D^{(35)}_7$ and in IV_7 are not compatible:

Class 1 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 0$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 241 \times 2^{24}$
Class 2 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 74 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 16 \times 2^{24}$
Class 3 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 58 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 122 \times 2^{24}$
Class 4 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 219 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 127 \times 2^{24}$
Class 5 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 237 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 195 \times 2^{24}$
Class 6 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 35 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 114 \times 2^{24}$
Class 7 $IV'_7 \oplus \overleftrightarrow{IV}_7 = 212 \times 2^{24}$, $D'^{(35)}_7 \oplus \overleftrightarrow{D}^{(35)}_7 = 199 \times 2^{24}$

B.3 Symmetry Classes for the Final Transformation

The final transformation is based on a slightly modified compression function and similar symmetry classes can be found. However, we note that all the messages and message pairs than can give a symmetric expanded message have a non zero value in the last message byte ($M[63]$ for SIMD-256, or $M[127]$ for SIMD-512). Since the message input of the final compression function is in fact the length of the message being hashed, this means that the message length must be at least 2^{504} for SIMD-256, and at least 2^{1016} for SIMD-512. Hashing such a long message is completely meaningless[3], so it is safe to say that the distinguisher cannot be used against the final transformation.

C Proof of Theorem 1

In this section we show that the prefix-free iteration of an ideal compression function is indifferentiable from a random oracle, thus proving Theorem 1. The content of this section borrows very much to the proof in the extended version of [7].

We consider a *simulator* $\mathcal{S}$, which has oracle access to a random oracle $RO : \{0,1\}^* \to \{0,1\}^p$, and whose task is to simulate a random compression function. The pseudo-code of the simulator is shown in Figure 5 page 28, but here are a few comments. The simulator maintains a log of the queries it has answered to. This knowledge is maintained under the form of a graph $G = (V, E)$, where the set of vertices V is a subset of $\mathcal{H}$, and where the edges are labelled by message blocks from $\mathcal{M}$. The semantic of this graph is that there is an edge labelled by m between h and h' if the simulator let the distinguisher know that $f(h, m) = h'$. We denote this by $h \xrightarrow{m} h'$. Initially,

[3]If it is feasible to hash these messages, then the hash function can be broken by brute force and does not offer any kind of security.

the graph contains only a single vertex IV. The simulator also maintains a subset of V denoted by Reach, consisting of the vertices that are reachable from IV. It also associates to each vertex v in Reach an *ancestor* in Reach. This allows to efficiently reconstruct the sequence of message blocks that maps IV to v, given v. We will note $IV \xrightarrow[*]{M} v$ when there is such a path between IV and v. At the beginning, Reach only contains the IV.

```
1: function FreshValue($(h, m)$)
2:     if $IV \xrightarrow[*]{M} h \in \mathsf{Reach}$ then
3:         if there exist $M'$ such that $M \| m = g(M')$ then
4:             $h' \leftarrow RO(M')$
5:         else
6:             $h' \xleftarrow{\$} \mathcal{H}$
7:         end if
8:         $\mathsf{Reach} \leftarrow \mathsf{Reach} \cup \left\{h \xrightarrow{m} h'\right\}$
9:     else
10:        $h' \xleftarrow{\$} \mathcal{H}$
11:    end if
12:    $V \leftarrow V \cup \{h, h'\}$
13:    $E \leftarrow E \cup \left\{h \xrightarrow{m} h'\right\}$
14:    return $h'$
15: end function

16: function Simulator($(h, m)$)
17:    if there exist a vertex $h' \in V$ and an edge $h \xrightarrow{m} h'$ in $E$ then
18:        return this $h'$
19:    else
20:        $h' \leftarrow$ FreshValue($(h, m)$)
21:        return $h'$
22:    end if
23: end function
```

Fig. 5. Pseudo-code of the Simulator $\mathcal{S}$

Let F be a random function. Now, a *distinguisher* $\mathcal{D}$ interacts with either H^F and F (we say that it is in the "construction world"), or with RO and $\mathcal{S}$ (and we say that it is in the "random oracle world"), and it has to tell in which world it is. More formally, $\mathcal{D}$ is a Turing machine that has two interfaces. It should output "1" if H^F and F are answering its oracle queries, and "0" if RO and $\mathcal{S}$ are. Our objective is to show that the following holds for a small ε:

$$\left| \mathbb{P}\left[\mathcal{D}^{H^F, F} = 1\right] - \mathbb{P}\left[\mathcal{D}^{RO, \mathcal{S}} = 1\right] \right| \leq \varepsilon$$

The proof uses a hybrid argument through a sequence of games. We will denote by q_S and q_O the number of queries sent to the Simulator and the Oracle respectively, by the distinguisher.

Game 1: The distinguisher is in the random oracle world. It has access to RO and $\mathcal{S}$. Let G_1 be the event that $\mathcal{D}$ outputs "1" in this setting:

$$\mathbb{P}\left[G_1\right] = \mathbb{P}\left[\mathcal{D}^{RO, \mathcal{S}} = 1\right]$$

Game 2: We introduce a dummy *relay algorithm* $\mathcal{T}$, which has oracle access to RO. Given a random oracle query from the distinguisher, $\mathcal{T}$ just send the query to RO, and transmits the answer of RO back to $\mathcal{D}$. Let G_2 be the event that $\mathcal{D}$ outputs "1" in this case. Since the view of $\mathcal{D}$ is left unchanged, we have:

$$\mathbb{P}\big[G_2\big] = \mathbb{P}\big[\mathcal{D}^{\mathcal{T}^{RO},\mathcal{S}} = 1\big] = \mathbb{P}\big[G_1\big]$$

Game 3: In this game, we modify the simulator $\mathcal{S}$. In particular, we restrict the responses of the simulator such that they never satisfy certain specific failure conditions. If the simulator comes up with a response that would result in an inconsistent state, then it fails explicitly instead of sending that response. The failure conditions describe certain situations that could be exploited by the distinguisher, such as collisions on the internal state. We just slightly change the FreshValue function:

```
1: function FreshValue($h, m$)
2:     if $IV \xrightarrow[*]{M} h \in \mathsf{Reach}$ then
3:         if there exist $M'$ such that $M \| m = g(M')$ then
4:             $h' \leftarrow RO(M')$
5:         else
6:             $h' \xleftarrow{\$} \mathcal{H}$
7:         end if
8:         if $h' \in V$ then
9:             Abort
10:        end if
11:        $\mathsf{Reach} \leftarrow \mathsf{Reach} \cup \left\{h \xrightarrow{m} h'\right\}$
12:    else
13:        $h' \xleftarrow{\$} \mathcal{H}$
14:    end if
15:    $V \leftarrow V \cup \{h, h'\}$
16:    $E \leftarrow E \cup \left\{h \xrightarrow{m} h'\right\}$
17:    return $h'$
18: end function
```

It should be clear that until no abort occur, the subgraph Reach is in fact a tree rooted in IV. This follows from the fact that the simulator aborts as soon as a collision in the internal state is detected. The new value h' is always drawn uniformly at random. It should be clear that as long as the simulator does not abort, the number of nodes in V is upper-bounded by $2q_S + 1$.

Therefore, for a given query, the probability of failure is upper-bounded by $(2q_S + 1)/2^p$. For all the q_S queries sent by the distinguisher, the probability of failure is therefore less than $q_S \cdot (2 \cdot q_s + 1)/2^p$. Let G_3 be the event that $\mathcal{D}$ outputs "1" in this case. Since the view of $\mathcal{D}$ only changes when the simulator aborts, we have:

$$\big|\mathbb{P}\big[G_3\big] - \mathbb{P}\big[G_2\big]\big| \leq 2 \cdot \frac{(q_S + 1)^2}{2^p}$$

Game 4: In this game, we modify the relay algorithm and leave the simulator unchanged. The underlying idea is to make the responses of the relay algorithm directly dependent on the simulator. Thus, instead of giving the new relay algorithm $\mathcal{T}_1$ an oracle access to the random oracle RO, it is now given oracle access to the simulator $\mathcal{S}_0$. On a random oracle query X, the relay algorithm $\mathcal{T}_1$ computes the prefix-free encoding of X, $g(X)$. It then applies the Merkle-Damgård construction to $g(X)$ and queries the simulator $\mathcal{S}_0$ to evaluate the compression function. Thus the relay algorithm $\mathcal{T}_1$ is essentially the same as the random oracle construction pf-MD, except that it is based on the simulator $\mathcal{S}_0$ instead of random function F. Let G_4 denote the event that the distinguisher $\mathcal{D}$ outputs "1" when given oracle access to $\mathcal{T}_1$ and $\mathcal{S}_0$ in this game. Thus, we know that

$$\mathbb{P}\big[G_4\big] = \mathbb{P}\big[\mathcal{D}^{\mathcal{T}_1,\mathcal{S}_0} = 1\big]$$

Before going further, we establish two key properties of $\mathcal{S}_0$. Let us consider the sequence $\mathcal{Q}$ of queries (h_i, m_i, h'_i) sent to $\mathcal{S}_0$, where h'_i is the answer and (h_i, m_i) is the question. We say that the IV is *reachable*, and at a given point in the simulation h'_i is reachable if there has been a previous query (h_i, m_i, h'_i) where h_i was reachable. Then:

i) Until $\mathcal{S}_0$ fails, Reach precisely describes the set of reachable chaining values.
ii) Until $\mathcal{S}_0$ fails, Reach describes a tree.

These two properties are easy to establish by induction on the number of queries. When the simulator detects that h_i is reachable, it puts its answer h'_i in Reach. What guarantees that our two properties hold is that $\mathcal{S}_0$ aborts if h'_i was already "known". Thus, the set of reachable values can only be extended by one element, namely h'_i, and Reach is updated accordingly.

Next, we claim that the following three statements hold:

i) In Game 3, *i.e.*, when $\mathcal{D}$ interacts with $\big(\mathcal{T}^{RO}, \mathcal{S}_0\big)$, the answers of $\mathcal{S}_0$ are consistent with those of RO as long as $\mathcal{S}_0$ does not abort.
ii) In Game 4, *i.e.*, when $\mathcal{D}$ interacts with $\Big(\mathcal{T}_1^{pf-MD(\mathcal{S}_0)}, \mathcal{S}_0\Big)$, the answers of $\mathcal{S}_0$ are consistent with those of RO as long as $\mathcal{S}_0$ does not abort.
iii) $\mathcal{T}^{RO}$ and $\mathcal{T}_1^{pf-MD(\mathcal{S}_0)}$ give the same answers until the simulator aborts.

From these three points, we can deduce that the view of the distinguisher $\mathcal{D}$ remains unchanged from game 3 to game 4 if the simulator $\mathcal{S}_0$ does not fail in either of the two games.

Proof. *i)* To detect an inconsistency between $\mathcal{S}_0$ and RO, one has to build a chain of queries corresponding to a valid message, and compare with the output of RO with the last query of the chain. Note that if the chain is built out-of-order, then the simulator will abort. Therefore the last query to be sent to $\mathcal{S}_0$ is the final block of the prefix free encoding of M. When $\mathcal{S}_0$ detects the final block of a message, it queries RO on the decoded message, which is unique because Reach is a tree. The answers of RO and $\mathcal{S}_0$ are thus consistent.
ii) The justification is the same as in the previous point. The fact that $\mathcal{T}_1$ sends extra queries does not change the fact that $\mathcal{S}_0$ answers are consistent with the Random Oracle.
iii) Since $\mathcal{S}_0$ is consistent with the VIL-RO, the relay algorithm $\mathcal{T}_1$ does in fact return $RO(M)$ by applying the pf-MD construction with $\mathcal{S}_0$. □

We can finally complete the argument by observing that if the maximum length of the prefix-free encoding of a random oracle query made by $\mathcal{D}$ is κ blocks, then,

$$\begin{aligned}\big|\mathbb{P}\big[G_4\big] - \mathbb{P}\big[G_3\big]\big| &\le \mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 3}\big] + \mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 4}\big] \\ &\le 2 \cdot \frac{(q_S+1)^2 + (q_S + \kappa \cdot q_O + 1)^2}{2^p}\end{aligned}$$

Game 5: In this game, we modify the simulator $\mathcal{S}_0$ so as to make its responses independent of the random oracle RO. For this purpose, we remove the random oracle RO from this game entirely and the new simulator $\mathcal{S}_1$ always chooses a random p-bit response itself, even in situations where $\mathcal{S}_0$ would have consulted RO. We also remove all the failure conditions from the new simulator $\mathcal{S}_1$. More precisely, we change the simulator in the following way:

```
1: function FreshValue$(h, m)$
2:     $h' \xleftarrow{\$} \mathcal{H}$
3:     $V \leftarrow V \cup \{h, h'\}$
4:     $E \leftarrow E \cup \left\{h \xrightarrow{m} h'\right\}$
5:     return $h'$
6: end function
```

The responses of these two simulators are identical apart from the failure conditions which are used by $\mathcal{S}_0$ and not by $\mathcal{S}_1$: even when $\mathcal{S}_0$ consults the random oracle, its response is still uniformly distributed. Thus, the distinguisher does not notice a difference between these games if in game 4, the simulator $\mathcal{S}_0$ does not fail.

Let G_5 denote the event that the distinguisher $\mathcal{D}$ outputs "1" in game 5, so that

$$\mathbb{P}\big[G_5\big] = \mathbb{P}\big[\mathcal{D}^{\mathcal{T}_1, \mathcal{S}_1} = 1\big]$$

Then we can deduce that:

$$\begin{aligned}\big|\mathbb{P}\big[G_5\big] - \mathbb{P}\big[G_4\big]\big| &\leq \mathbb{P}\big[\mathcal{S}_0 \text{ fails in game } 4\big] \\ &\leq 2 \cdot \frac{\left(q_S + \kappa \cdot q_O + 1\right)^2}{2^p}\end{aligned}$$

Game 6: This is the final game of our argument. Here we finally replace the simulator $\mathcal{S}_1$ with the random function F. Since the relay algorithm $\mathcal{T}_1$ simply implemented the prefix-free Merkle-Damgård construction, then the view of the distinguisher is in fact the construction world.

Now, by combining games 1 to 6, we can show that

$$\left|\mathbb{P}\big[\mathcal{D}^{H^F, F} = 1\big] - \mathbb{P}\big[\mathcal{D}^{RO, \mathcal{S}} = 1\big]\right| \leq 4 \cdot \frac{\left(q_S + 1\right)^2 + \left(q_S + \kappa \cdot q_O + 1\right)^2}{2^p}$$

D Proof of Theorem 2

The proof proceed in the same way as the proof of Theorem 1, given in Appendix C. The simulator is shown in Figure 3 page 12. The pseudo-code shows $\mathcal{S}_0$ with the failure conditions. Before going further, a few comments on $\mathcal{S}_0$ are in order. When it receives a query (h, m), it checks whether there exist a *symmetric query* $\left(\overline{h}, \overline{m}\right)$, that would trigger the symmetry relation (*i.e.*, it checks whether the node (h, m) is connected to something in $G'_{\mathcal{R}_2}$). If such a query exist, then both are treated simultaneously in a "symmetric" way. In particular, if either one of these concerns a reachable state, then it is treated specially, even if it not the original query, but the "symmetric" one.

Game 3: Let us discuss the probability that $\mathcal{S}_0$ fails. It can only happen if h is reachable, which in turn means that h' is randomly distributed in $\mathcal{H}$. $\mathcal{S}_0$ aborts when $h' \in \mathcal{W}$, $h' \in V$ or $\mathsf{Reach} \cup \{h'\}$ covers an edge of $G_{\mathcal{R}_2}$. The probability that $h' \in \mathcal{W}$ is $|\mathcal{W}|/2^p$, and the probability that $h' \in V$ is upper-bounded by $(4 \cdot q_S + 1)/2^p$, since $|V| \leq 4 \cdot q_s + 1$. Let us now discuss the probability that an edge of $G_{\mathcal{R}_2}$ is covered by Reach.

A simple induction on the number of queries shows that the chaining values in Reach are all randomly and independently distributed in $\mathcal{H}$ (this is because Reach is always extended by h' on line 19, and h' is itself always generated randomly). If we ignore the abort conditions, Reach is a random subset of $\mathcal{H}$ of size $k \leq q_s + 1$ after q_S queries. There are $\binom{2^p}{k}$ such subsets, and amongst these $\binom{2^p}{k-2}$ cover a given edge. The probability that at least one edge out of $|\mathcal{WP}|$ is covered is thus upper-bounded by $|\mathcal{WP}| \cdot \binom{2^p}{k-2} / \binom{2^p}{k}$, which it itself upper-bounded by $|\mathcal{WP}| \cdot k^2/(2^p-k)^2$. After q_s queries, the probability of failure is therefore bounded by:

$$\big|\mathbb{P}\big[G_3\big] - \mathbb{P}\big[G_2\big]\big| \leq 4 \cdot \frac{\left(q_S+1\right)^2}{2^p} + |\mathcal{W}| \cdot \frac{q_S+1}{2^p} + |\mathcal{WP}| \cdot \frac{q_S+1^2}{\left(2^p-q_S-1\right)^2}$$

Game 4: We claim that the following four statements hold:

i) In Game 3, *i.e.*, when $\mathcal{D}$ interacts with $\left(\mathcal{T}^{RO}, \mathcal{S}_0\right)$, the answers of $\mathcal{S}_0$ are consistent with those of *RO* as long as $\mathcal{S}_0$ does not abort.

ii) In Game 4, *i.e.*, when $\mathcal{D}$ interacts with $\left(\mathcal{T}_1^{pf-MD(\mathcal{S}_0)}, \mathcal{S}_0\right)$, the answers of $\mathcal{S}_0$ are consistent with those of *RO* as long as $\mathcal{S}_0$ does not abort.

iii) $\mathcal{T}^{RO}$ and $\mathcal{T}_1^{pf-MD(\mathcal{S}_0)}$ give the same answers until the simulator aborts.

iv) As long as it does not abort, the answer of $\mathcal{S}_0$ always comply with the relation $\mathcal{R}$.

Consistency with the VIL-RO. Establishing the first three points can be done in the same as it was done in the proof of Theorem 1. The simulator relies on the fact that Reach is a tree, and that it exactly describes the reachable chaining values in V. This can be established by arguing that if $\mathcal{S}$ does not abort, then h' is the only new reachable chaining value created by the current invocation of FreshValue. Note that $\overline{h}$, if it exists, is not reachable.

Conformance to the Relation. The main point is that the relation can never be falsified on reachable states, and that the samplers are used on non-reachable states to ensure that the answers are consistent with the relation. More precisely, the simulator aborts as soon as a state in $\mathcal{W}$ becomes reachable, or a pair of states in $\mathcal{WP}$ becomes reachable.

Let us assume that the distinguisher can find a query (h, m, h') with $h \xrightarrow{m} h'$ such that $\mathcal{R}_1(h, m, h')$ does not hold. Then we have $h \in \mathcal{W}$ by definition of $\mathcal{W}$, therefore h cannot be reachable and h' has necessarily been build by $\mathbf{Sampler}_1$. By definition of $\mathbf{Sampler}_1$, $\mathcal{R}_1(h, m, h')$ must hold.

Similarly, let us assume that the distinguisher finds $h_1 \xrightarrow{m_1} h_1'$ and $h_2 \xrightarrow{m_2} h_2'$ such that $\mathcal{R}_2(h_1, m_1, h_2, m_2, h_1', h_2')$ does not hold. By definition of $\mathcal{WP}$ we have $(h_1, h_2) \in \mathcal{WP}$ therefore h_1 and h_2 cannot both be reachable. Moreover, we have $(h_1, m_1) \leftrightarrow (h_2, m_2) \in G'_{\mathcal{R}_2}$. Without loos of generality, we assume that h_2 is not reachable. When the first query involving (h_1, m_1) or (h_2, m_2) was sent to $\mathcal{S}_0$, the simulator built the second query. If h_1 was reachable, h_1' has been built by calling the VIL-RO and h_2' has been built by $\mathbf{Sampler}_2$, with assures that $\mathcal{R}_2(h_1, m_1, h_2, m_2, h_1', h_2')$ holds. Similarly, if h_1 is not reachable, h_1' has been built by $\mathbf{Sampler}_1$, and h_2' by $\mathbf{Sampler}_2$. We note that if h_1 was not reachable at the time when it was queried, it cannot become reachable later without causing the simulator to abort.

Finally, we obtain that the view of the distinguisher does not change as long as the simulator does not abort:

$$\begin{aligned}\big|\mathbb{P}\big[G_4\big] - \mathbb{P}\big[G_3\big]\big| &\leq \mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 3}\big] + \mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 4}\big] \\ &\leq 2 \cdot \mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 4}\big] \\ &\leq 8 \cdot \frac{\left(q_S + \kappa \cdot q_O\right)^2}{2^p} + 2|\mathcal{W}| \cdot \frac{q_S + \kappa \cdot q_O}{2^p} + 2|\mathcal{WP}| \cdot \frac{\left(q_S + \kappa \cdot q_O\right)^2}{\left(2^p - q_S - \kappa \cdot q_O\right)^2}\end{aligned}$$

And we conclude:

$$\begin{aligned}
\Big|\mathbb{P}\big[\mathcal{D}^{H^F,F}=1\big]-\mathbb{P}\big[\mathcal{D}^{RO,\mathcal{S}}=1\big]\Big| &\leq \big|\mathbb{P}\big[G_1\big]-\mathbb{P}\big[G_6\big]\big| \\
&\leq 4\mathbb{P}\big[\mathcal{S}_0 \text{ fails in Game 4}\big] \\
&\leq 16\cdot\frac{\left(q_S+\kappa\cdot q_O\right)^2}{2^p}+4\cdot|\mathcal{W}|\cdot\frac{q_S+\kappa\cdot q_O}{2^p} \qquad +4\cdot|\mathcal{WP}|\cdot\frac{\left(q_S+\kappa\cdot q_O\right)^2}{\left(2^p-q_S-\kappa\cdot q_O\right)^2}
\end{aligned}$$