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. 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. 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.
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 [1,9] 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

*The full version of this paper appears as IACR ePrint report 2010/323 [5].

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.

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 [13] 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.

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 $\mathbb{F}_{257}$) to double the size of the message. The NTT is actually used as a Reed-Solomon code.
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.

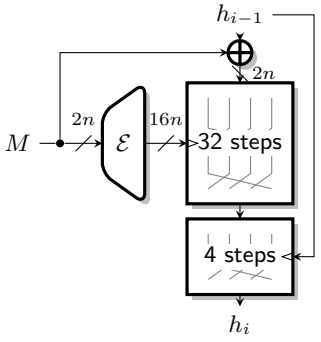

Fig. 1. SIMD modified Davies-Meyer mode

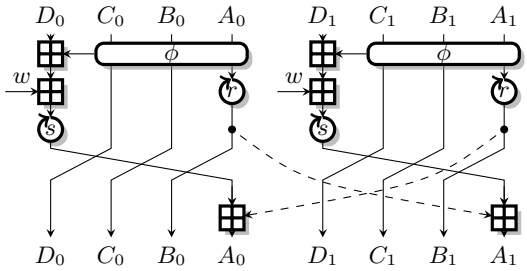

Fig. 2. SIMD compression rounds. There are 4 parallel Feistels in SIMD-256, and 8 parallel Feistels in SIMD-512.

4. Permute the 16-bit words and pack them into 32-bit words.

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].

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 [1,9] 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 [11, Table 1.1]. Because of this property, if we have

[1]This design choice was guided by implementation efficiency

$y_i = y_{i\oplus 2}$ 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.

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

$$\overleftrightarrow{(a,b,c,d)} = (b,a,d,c) \qquad \underset{\leftrightarrow}{(a,b,c,d)} = (c,d,a,b) \qquad \underset{\leftrightarrow}{\overleftrightarrow{(a,b,c,d)}} = (d,c,b,a)$$

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):

$$y_i = y'_{i\oplus 2} \Leftrightarrow \mathcal{E}(M) = \overleftrightarrow{\mathcal{E}(M')} \qquad y_i = y'_{i\oplus 4} \Leftrightarrow \mathcal{E}(M) = \underset{\leftrightarrow}{\mathcal{E}(M')}$$
$$y_i = y'_{i\oplus 6} \Leftrightarrow \mathcal{E}(M) = \underset{\leftrightarrow}{\overleftrightarrow{\mathcal{E}(M')}}$$

By solving the corresponding linear systems, we can compute the sets of symmetric messages (the sets are described in the full version of this paper). We can count the symmetric messages M such that $\mathcal{E}(M) = \overleftrightarrow{\mathcal{E}(M)}$, and the pairs of messages M, M' such that $\mathcal{E}(M) = \overleftrightarrow{\mathcal{E}(M')}$:

Sym. class (SIMD-256)		msg	pairs
$\overleftrightarrow{\bullet}$	$y_i = y'_{i\oplus 2}$ $W_i = W'_{i\oplus 1}$	2^8	$256 \cdot 255$
$\underset{\leftrightarrow}{\bullet}$	$y_i = y'_{i\oplus 4}$ $W_i = W'_{i\oplus 2}$	2^{16}	$(256 \cdot 255)^2$
$\underset{\leftrightarrow}{\overleftrightarrow{\bullet}}$	$y_i = y'_{i\oplus 6}$ $W_i = W'_{i\oplus 3}$	2^8	$256 \cdot 255$

Sym. class (SIMD-512)		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	$256 \cdot 255$

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.

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. A concrete example of message and chaining value is given in the full version of this paper.

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. 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):

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

The corresponding weak states are:

$$\mathcal{W}_1^M := \{x \oplus M \mid x = \overleftrightarrow{x}\} \qquad \mathcal{W}_2^{M,M'} := \left\{(h, \overleftrightarrow{h} \oplus M' \oplus \overleftrightarrow{M})\right\}$$

The study of the symmetry classes of SIMD shows that:

$\lvert\mathcal{W}_1\rvert \approx 2^{256} \cdot 2^{16}$	$\lvert\mathcal{W}_1\rvert \approx 2^{512} \cdot 2^{32}$	for SIMD-512
$\lvert\mathcal{W}_2\rvert < 2^{512} \cdot 2^{32}$	$\lvert\mathcal{W}_2\rvert < 2^{1024} \cdot 2^{64}$	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 the full version of the paper 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 [1,9].
- 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 [1,9]. 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.

To summarize, reaching a symmetric state in SIMD is far less interesting than reaching a symmetric state in CubeHash or in Lesamnta. Table 1 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

Table 1. 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}

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 [12,8], 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 [12] 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 $(t_{\mathcal{D}}, t_{\mathcal{S}}, q_S, q_0, \varepsilon)$-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_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 [6]. 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, $g : \{0,1\}^* \to \mathcal{M}^*$ 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 [8]. 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 [1,9], and SIMD (described in Section 2).

[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.

– 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 [6], we model this situation by saying that there are two relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that:

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

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:

$$\mathcal{WP} = \left\{h_1 \leftrightarrow h_2 \mid \exists m_1, m_2, h_1', h_2' \in \mathcal{M}^2 \times \mathcal{H}^2 \text{s.t.} \mathcal{R}_2(h_1,m_1,h_2,m_2,h_1',h_2') = 0\right\}$$

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:

$$\mathcal{WP}' = \left\{(h_1,m_1) \leftrightarrow (h_2,m_2) \mid \exists h_1', h_2' \in \mathcal{H}^2 \text{s.t.} \mathcal{R}_2(h_1,m_1,h_2,m_2,h_1',h_2') = 0\right\}$$

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 $\mathbf{Sampler}_1$ and $\mathbf{Sampler}_2$ such that:

Sampler$_1(h, m)$
$h' \xleftarrow{\$} \{f(h, m) \mid f \in \mathcal{F}[\mathcal{R}]\}$; return h'
Sampler$_2(h_1, m_1, h_2, m_2, h'_1)$
$h'_2 \xleftarrow{\$} \{f(h_2, m_2) \mid f \in \mathcal{F}[\mathcal{R}]$ 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 $2^{n/2}$.

The proof is heavily based on the proof in the extended version of [8]. Due to space constraints, the proof is not included in this paper, but can be found in the full version.

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.* [15]. 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

The minimal distance of the message expansion of SIMD is at least 520. This distance counts the number of active bits, but we can also show that even if consecutive bits can collapse to give a single signed difference, we still have a minimal distance of 455 (respectively 903 for SIMD-512). The only case where adjacent differences can collapse to give a smaller signed difference is when the bits 15 and 16 are active in the two 16-bit words that are packed into a 32-bit word. In Section 4.4, we disregard this property and we just consider that the message introduces 520 differences through the message expansion, but the model used in Section 4.5 accounts precisely 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 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.

$\boxplus$-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 [14]: 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.

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 [7,15]. 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 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 $520/4.5 \approx 115.55$.

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

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.

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.

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.

A simplified version of the ILP is given in the full version of this paper. 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.

Therefore, we expect that the best differential path in SIMD is much worse that our lower bound.

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. Bouillaguet, C., Fouque, P.A., Leurent, G.: Security analysis of simd. Cryptology ePrint Archive, Report 2010/323 (2010) `http://eprint.iacr.org/`.
6. 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/`.
7. 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
8. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: CRYPTO'05. (2005) 430–448
9. 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/`.
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. 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
13. 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
14. 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
15. 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