Higher-Order Differential Meet-in-The-Middle Preimage Attacks on SHA-1 and BLAKE

Thomas Espitau[1,2], Pierre-Alain Fouque[3,4], and Pierre Karpman[2,5⋆]

[1] École normale supérieure de Cachan, France
[2] Inria, France
[3] Université de Rennes 1, France
[4] Institut Universitaire de France, France
[5] Nanyang Technological University, Singapore

tespitau@ens-cachan.fr, pa.fouque@gmail.com, pierre.karpman@inria.fr

Abstract. At CRYPTO 2012, Knellwolf and Khovratovich presented a differential formulation of advanced meet-in-the-middle techniques for preimage attacks on hash functions. They demonstrated the usefulness of their approach by significantly improving the previously best known attacks on SHA-1 from CRYPTO 2009, increasing the number of attacked rounds from a 48-round *one-block pseudo-preimage without padding* and a 48-round *two-block preimage without padding* to a 57-round *one-block preimage without padding* and a 57-round *two-block preimage with padding*, out of 80 rounds for the full function. In this work, we exploit further the differential view of meet-in-the-middle techniques and generalize it to higher-order differentials. Despite being an important technique dating from the mid-90's, this is the first time higher-order differentials have been applied to meet-in-the-middle preimages. We show that doing so may lead to significant improvements to preimage attacks on hash functions with a simple linear message expansion. We extend the number of attacked rounds on SHA-1 to give a 62-round *one-block preimage without padding*, a 56-round *one-block preimage with padding*, and a 62-round *two-block preimage with padding*. We also apply our framework to the more recent SHA-3 finalist BLAKE and its newer variant BLAKE2, and give an attack for a 2.75-round *preimage with padding*, and a 7.5-round *pseudo-preimage on the compression function*.

Keywords: Hash function, preimage attack, higher-order differential meet-in-the-middle, SHA-1, BLAKE, BLAKE2

1 Introduction

A hash function is a cryptographic primitive that is used to compress any binary string of arbitrary length to one of a fixed predetermined length: $H : \{0,1\}^* \to \{0,1\}^n$. Hash functions hold a special role among cryptographic primitives, as operating without a key. This makes the analysis of their security somewhat harder than for most other primitives, but three notions are commonly used for that purpose: *collision resistance*, means that it is hard for an attacker to find two distinct strings (or messages) m and m' such that $H(m) = H(m')$; *second preimage resistance* means that it is hard given a predetermined message m to find a distinct message m' such that $H(m) = H(m')$; and *preimage resistance* means that it is hard given a target t to find a message m such that $H(m) = t$. The hardness level associated with these three notions depends on the length of the output of H, and is $\mathcal{O}(2^{\frac{n}{2}})$ for collision resistance, and $\mathcal{O}(2^n)$ for (second) preimage resistance. In addition to these notions, it is also common to evaluate the security of hash function through the one of its building blocks.

⋆ Partially supported by the Direction Générale de l'Armement and by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06)

In this work, we give a framework that can be used to attack the preimage resistance of hash functions designed around certain principles. We show the usefulness of our approach by improving the best known attacks on two popular hash functions: the first is the NIST standard SHA-1 [15], which is a widely used function originally designed by the NSA in the 1990's; the second is the SHA-3 finalist BLAKE [3], which along with its updated version BLAKE2 is increasingly being used in modern applications.

Our starting point is the meet-in-the-middle technique, which was first used in cryptography by Diffie and Hellman in 1977 to attack double-encryption [7]. Its use for preimage attack is much more recent and is due to Aoki and Sasaki, who used it as a framework to attack various hash functions, including for instance SHA-0 and SHA-1 [2]. The basic principle behind a meet-in-the-middle technique is to exploit the fact that some value can be computed in two different ways involving different parts of a secret, which can then be sampled independently of each other. In the case of hash function cryptanalysis, there is no actual secret to consider, but a similar technique can nonetheless be exploited in certain cases; we show in more details how to do so in the preliminaries of §2.

At CRYPTO 2012, Knellwolf and Khovratovich introduced a differential formulation of the meet-in-the-middle framework of Aoki and Sasaki, which they used to improve the best attacks on SHA-1. One of the main interests of their approach is that it simplifies the formulation of several advanced extensions of the meet-in-the-middle technique, and thereby facilitates the search for attack parameters (in the case of meet-in-the-middle attacks, this roughly corresponds to finding good partitions for the 'secret').

In this work, we further exploit this differential formulation and generalize it to use higher-order differentials, which were introduced in cryptography by Lai in 1994 [12]. The essence of this technique is to consider 'standard' differential cryptanalysis as exploiting properties of the first-order derivative of the function one wishes to analyze; it is then somehow natural to generalize the idea and to consider higher-order derivatives as well. Let us illustrate this with a small example using XOR '$\oplus$' differences: consider a function f and assume the differential $\Delta_\alpha f \triangleq f(x) \oplus f(x \oplus \alpha) = A$ holds with a good probability; this is the same as saying that the derivative of f in α is biased towards A. In particular, if f is linear, this is equal to a constant value $f(\alpha)$, though this is obviously not true in general. Now consider the value $\Delta_\alpha f(x) \oplus \Delta_\alpha f(x \oplus \beta) = f(x) \oplus f(x \oplus \alpha) \oplus f(x \oplus \beta) \oplus f(x \oplus \alpha \oplus \beta)$, which corresponds to taking the derivative of f twice, first in α, and then in β. The nice point about doing this is that this function may be more biased than $\Delta_\alpha f$ was, for instance by being constant when $\Delta_\alpha f$ is linear. This process can be iterated at will, each time decreasing the algebraic degree of the resulting function until it reaches zero.

As higher-order differentials are obviously best formulated in differential form, they combine neatly with the differential view of the framework of Knellwolf and Khovratovich, whereas using such a technique independently of any differential formulation would probably prove to be much more difficult. As a final motivation for this generalization, we show a small application to the analysis of the MD4 hash function [17]. This does not improve the best known preimage attacks [9,20], but gives a good illustration of the potential of this technique.

Higher-order differentials for the compression function of MD4. The inverse of the state update function inside the compression function of MD4 is of the form: $q_{i-4} \leftarrow\!\shortmid (q_i \lll s_i) - \Phi(q_{i-1}, q_{i-2}, q_{i-3}) - m_j$, with Φ a bitwise Boolean function, $\lll$ the operator for bitwise rotation, and the addition being done modulo 2^{32}. Four consecutive steps of this inverse function can thus be written as:

$$q_3 \leftarrow\!\shortmid (q_7 \lll s_7) - \Phi(q_6, q_5, q_4) - m_7$$

$$q_2 \leftarrow\!\shortmid (q_6 \lll s_6) - \Phi(q_5, q_4, q_3) - m_6$$

$$q_1 \leftarrow\!\shortmid (q_6 \lll s_5) - \Phi(q_4, q_3, q_2) - m_5$$

$$q_0 \leftarrow\!\shortmid (q_4 \lll s_4) - \Phi(q_3, q_2, q_1) - m_4$$

If we consider order-2 differentials on m_6 and m_7, with (additive, modulo 2^{32}) differences concentrated around the most significant bit, that is of the form `xxxx----` (an '`x`' denoting the presence of a difference, a '`-`' its absence, and a '`?`' an unknown condition), computing the state update from above on these differences results in a state $q_{0\ldots3}$ with differences of the form:

q_3^*: `?????-----`	$q_3^\star$: `----------`	$q_3^\dagger$: equal to q_3^*
q_2^*: `?????-----`	$q_2^\star$: `?????-----`	$q_2^\dagger$: `?????-----`
q_1^*: `?????-----`	$q_1^\star$: `?????-----`	$q_1^\dagger$: `?????-----`
q_0^*: `?????-----`	$q_0^\star$: `?????-----`	$q_0^\dagger$: `?????-----`
For diff. on m_7	For diff. on m_6	For diff. on m_6 & m_7

Thus there is no differences on the value $q_3^\dagger - q_3^\star - q_3^* + q_3$, nor on $q_2^\dagger - q_2^\star - q_2^* + q_2$ as can be seen from a simple computation. Hence one obtains a state with no differences on two out of the four state words with probability one; the differences on the remaining words also have a high probability. This good differential behaviour can then be exploited in a later attack[1].

Previous and new results on SHA-1 and BLAKE(2). The previous best preimage attacks on SHA-1 are due to Knellwolf and Khovratovich [11]. In their paper, they attack reduced versions of the function up to 52 steps for *one-block preimages with padding*, 57 steps for *one-block preimages without padding*, and 60 steps for *one-block pseudo-preimages with padding*. The latter two attacks can be combined to give 57 steps *two-block preimages with padding*. In this work, we present *one-block preimages with padding* up to 56 steps, *one-block preimages without padding* up to 62 steps, *one-block pseudo preimages with padding* up to 64 steps, resulting in *two-block preimages with padding* up to 62 steps.

The previous best known result for the BLAKE hash function, as far as preimages are concerned, is a 2.5-round attack by Li and Xu [13]. In a compression function model, the previous best attack reached 4 rounds [19]. For BLAKE2, the only known result is a pseudo-preimage attack on the full compression function targeting a small class of weak preimages of a certain form [8]. In this paper, we give a 2.75-round (resp. 3-round) *preimage attack* on BLAKE-512 and BLAKE2b, and a 7.5-round (resp. 6.75) pseudo-preimage on the compression functions of the larger (resp. smaller) variants of BLAKE and BLAKE2

We give a summary of these results in Table 1.

[1] In the very case of MD4, one can also use very good local collisions from order-1 message differences, and higher-order differentials do not typically outperform these; we just gave them for illustration

Table 1. Existing and new results on SHA-1 and BLAKE(2) (the complexity is given in base-2 logarithm).

Function	# blocks	# rounds	complexity	ref.
SHA-1	1	52	158.4	[11]
	1	52	156.7	§4.3
	1	56	159.4	§4.3
	2	57	158.8	[11]
	2	58	157.9	§4.4
	2	62	159.3	§4.4
SHA-1, without padding	1	57	158.7	[11]
	1	58	157.4	§4.2
	1	62	159	§4.2
SHA-1, pseudo-preimage	1	60	157.4	[11]
	1	61	156.4	§4.4
	1	64	158.7	§4.4
BLAKE-512	1	2.5	481	[13]
	1	2.75	510.3	§5.3
BLAKE2b	1	2.75	511	§5.3
BLAKE-256 c.f., pseudo-preimage	1	6.75	253.9	§5.2
BLAKE-512 c.f., pseudo-preimage	1	7.5	510.3	§5.2
BLAKE2s c.f., pseudo-preimage	1	6.75	253.8	§5.2
BLAKE2b c.f., pseudo-preimage	1	12	0 (weak class)	[8]
	1	7.5	510.3	§5.2

2 Meet-in-The-Middle Attacks and the Differential Framework from CRYPTO 2012

As a preliminary, we give a description of the meet-in-the-middle framework for preimage attacks on hash functions, and in particular of the differential formulation of Knellwolf and Khovratovich from CRYPTO 2012 [11].

The relevance of meet-in-the-middle for preimage attacks comes from the fact that many hash functions are built from a compression function which is an *ad hoc* block cipher used in one of the PGV modes [16]. One such popular mode is the so-called *Davies-Meyer*, where a compression function $h : \{0,1\}^v \times \{0,1\}^n \to \{0,1\}^v$ compressing a chaining value c with a message m to form the updated chaining value $c' = h(c,m)$ is defined as $h(c,m) \triangleq f(m,c) + c$, with $f : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^v$ a block cipher of key-length and message-length n and v respectively.

Given such a compression function, the problem of finding a preimage of t for h is equivalent to finding a key m for f such that $f(m,p) = c$ for a pair (p,c), with $c = t - p$. Additional constraints can also be put on p, such as prescribing it to a fixed initialization value iv.

In its most basic form, a meet-in-the-middle attack can speed-up the search for a preimage if the block cipher f can equivalently be described as the composition $f_2 \circ f_1$ of two block ciphers $f_1 : \mathcal{K}_1 \times \{0,1\}^v \to \{0,1\}^v$ and $f_2 : \mathcal{K}_2 \times \{0,1\}^v \to \{0,1\}^v$ with independent key spaces $\mathcal{K}_1, \mathcal{K}_2 \subset \{0,1\}^n$. Indeed, if this is the case, an attacker can select a subset $\{k_i^1, i = 1 \dots N_1\}$ (resp. $\{k_j^2, j = 1 \dots N_2\}$) of keys of $\mathcal{K}_1$ (resp. $\mathcal{K}_2$),

which together suggest $N \triangleq N_1 \cdot N_2$ candidate keys $k_{ij}^{12} \triangleq (k_i^1, k_j^2)$ for f by setting $f(k_{ij}^{12}, \cdot) = f_2(k_j^2, \cdot) \circ f_1(k_i^1, \cdot)$.

Since the two sets $\{f_1(k_i^1, p), i = 1 \dots N_1\}$ and $\{{f_2}^{-1}(k_j^2, c), j = 1 \dots N_2\}$ can be computed independently, the complexity of testing $f(k_{ij}^{12}, p) = c$ for N keys is only of $\mathcal{O}(\max(N_1, N_2))$ time and $\mathcal{O}(\min(N_1, N_2))$ memory, which is less than N and can be as low as $\sqrt{N}$ when $N_1 = N_2$.

2.1 Formalizing meet-in-the-middle attacks with related-key differentials

Let us denote by $(\alpha, \beta) \xrightarrow[p]{f} \gamma$ the fact that $\Pr_{(x,y)}\left[f(x \oplus \alpha, y \oplus \beta) = f(x, y) \oplus \gamma\right] = p$, meaning that (α, β) is a related-key differential for f that holds with probability p. The goal of an attacker is now to find two linear sub-spaces D_1 and D_2 of $\{0,1\}^m$ such that:

$$D_1 \cap D_2 = \{0\} \tag{1}$$

$$\forall \delta_1 \in D_1 \; \exists \; \Delta_1 \in \{0,1\}^v \text{ s.t. } (\delta_1, 0) \xrightarrow[1]{f_1} \Delta_1 \tag{2}$$

$$\forall \delta_2 \in D_2 \; \exists \; \Delta_2 \in \{0,1\}^v \text{ s.t. } (\delta_2, 0) \xrightarrow[1]{{f_2}^{-1}} \Delta_2. \tag{3}$$

Let d_1 and d_2 be the dimension of D_1 and D_2 respectively. Then for a set M of messages $\mu_i \in \{0,1\}^m$ (or more precisely the quotient space of $\{0,1\}^m$ by $D_1 \oplus D_2$), one can define $\#M$ distinct sets $\mu_i \oplus D_1 \oplus D_2$ of dimension $d_1 + d_2$ (and size $2^{d_1+d_2}$), which can be tested for a preimage with a complexity of only $\mathcal{O}(\max(2^{d_1}, 2^{d_2}))$ time and $\mathcal{O}(\min(2^{d_1}, 2^{d_2}))$ memory. We recall the procedure to do so in Alg. 1.

Algorithm 1: Testing $\mu \oplus D_1 \oplus D_2$ for a preimage [11]

Input: $D_1, D_2 \subset \{0,1\}^m$, $\mu \in \{0,1\}^m$, p, c
Output: A preimage of $c + p$ if there is one in $\mu \oplus D_1 \oplus D_2$, $\perp$ otherwise
Data: Two lists L_1, L_2 indexed by δ_2, δ_1 respectively

```
forall the δ2 ∈ D2 do
    L1[δ2] ↤ f1(μ ⊕ δ2, p) ⊕ Δ2
forall the δ1 ∈ D1 do
    L2[δ1] ↤ f2^{-1}(μ ⊕ δ1, c) ⊕ Δ1
forall the (δ1, δ2) ∈ D1 × D2 do
    if L1[δ2] = L2[δ1] then
        return μ ⊕ δ1 ⊕ δ2
return ⊥
```

Analysis of Alg. 1. For the sake of simplicity we assume that $d_1 = d_2 \triangleq d < \frac{v}{2}$. The running time of every loop of Alg. 1 is therefore $\mathcal{O}(2^d)$ (assuming efficient data structures and equality testing for the lists), and $\mathcal{O}(2^d)$ memory is necessary for storing L_1 and L_2. It is also clear that if the condition $L_1[\delta_2] = L_2[\delta_1]$ is met, then $\mu \oplus \delta_1 \oplus \delta_2$ is a preimage of $c+p$. Indeed, this translates to $f_1(\mu \oplus \delta_2, p) \oplus \Delta_2 = {f_2}^{-1}(\mu \oplus \delta_1, c) \oplus \Delta_1$, and using the differential properties of D_1 and D_2 for f_1 and f_2, we have that $f_1(\mu \oplus \delta_1 \oplus \delta_2, p) = f_1(\mu \oplus \delta_2, p) \oplus \Delta_1$ and ${f_2}^{-1}(\mu \oplus \delta_1 \oplus \delta_2, c) = {f_2}^{-1}(\mu \oplus \delta_1, c) \oplus \Delta_2$. Hence, $f_1(\mu \oplus \delta_1 \oplus \delta_2, p) = {f_2}^{-1}(\mu \oplus \delta_1 \oplus \delta_2)$, and $f(\mu \oplus \delta_1 \oplus \delta_2, p) = c$. This algorithm

therefore allows to search through 2^{2d} candidate preimages with a complexity of $\mathcal{O}(2^d)$, and thus gives a speed-up of 2^d. The complexity of a full attack is hence $\mathcal{O}(2^{v-d})$.

Comparison with basic meet-in-the-middle. When setting $\Delta_1 = \Delta_2 = 0$, this differential variant of the meet-in-the-middle technique becomes a special case of the general formulation of the basic technique given above: the key spaces $\mathcal{K}_1$ and $\mathcal{K}_2$ now possess a structure of affine spaces. The advantage of this restriction comes from the fact that it gives a practical way of searching for the key spaces, as differential path search is a well-studied area of symmetric cryptanalysis. Another major advantage is that it makes the formulation of several extensions to this basic attack very natural, without compromising the ease of the search for the key spaces. One such immediate extension is obviously to consider non-zero values for Δ_1 and Δ_2. As noted by Knellwolf and Khovratovich, this already corresponds to an advanced technique of *indirect matching* in the original framework of Aoki and Sasaki. Further extensions are detailed next.

2.2 Probabilistic truncated differential meet-in-the-middle

There are two natural ways to generalize the differential formulation of the meet-in-the-middle, which both correspond to relaxing one of the conditions from above. First, one can consider differentials of probability less than one (though a high probability is still usually needed); second, one can consider truncated differentials by using an equivalence relation '$\equiv$' instead of the equality (usually taken as a truncated equality: $a \equiv b[m] \Leftrightarrow a \wedge m = b \wedge m$ for $a, b, m \in \{0,1\}^v$), denoting by $(\alpha, \beta) \overset{f}{\underset{p}{\rightsquigarrow}} \gamma$ the fact that $\Pr_{(x,y)}\left[f(x \oplus \alpha, y \oplus \beta) \equiv f(x,y) \oplus \gamma\right] = p$. Hence equation 2 becomes:

$$\forall \delta_1 \in D_1 \; \exists \; \Delta_1 \in \{0,1\}^v \text{ s.t. } (\delta_1, 0) \overset{f_1}{\underset{p_1}{\rightsquigarrow}} \Delta_1, \tag{4}$$

for some probability p_1 and relation $\equiv$, and similarly for equation 3.

Again, these generalizations correspond to advanced techniques of Aoki and Sasaki's framework, which find here a concise and efficient description.

The only change to Alg. 1 needed to accommodate these extensions is to replace the equality by the appropriate equivalence relation on line 6. However, the fact that this equivalence holds no longer ensures that $\mu \oplus \delta_1 \oplus \delta_2$ is a preimage, which implies an increased complexity: firstly, even when it *is* a preimage, the relation on line 6 might not hold with probability $1 - p_1 p_2$, meaning that on average one needs to test $1/p_1 p_2$ times more candidates in order to account for the false negatives; secondly, if we denote by s the average size of the equivalence classes under $\equiv$ (when using truncation as above, this is equal to 2^{v-r} with r the Hamming weight of m), then on average one needs to check s potential preimages as returned on line 6 before finding a valid one, in order to account for the false positives. The total complexity of an attack with the modified algorithm is therefore $\mathcal{O}((2^{v-d} + s)/\tilde{p_1}\tilde{p_2})$, where $\tilde{p_1}$ and $\tilde{p_2}$ are the respective average probabilities for p_1 and p_2 over the spaces D_1 and D_2.

2.3 Splice-and-cut, initial structures and bicliques

These two techniques are older than the framework of [11], but are fully compatible with its differential approach.

Splice-and-cut was introduced by Aoki and Sasaki in 2008 [1]. Its idea is to use the feedforward of the compression function so as to be able to start the computation of f_1 and ${f_2}^{-1}$ not from p and c but from an intermediate value from the middle of the computation of f. If one sets $f = f_3 \circ f_2 \circ f_1$ and calls s the intermediate value ${f_3}^{-1}(c)$ (or equivalently $f_2 \circ f_1(p)$), an attacker may now sample the functions $f_1(t - f_3(s))$ and ${f_2}^{-1}(s)$ on their respective (as always independent) key-spaces when searching a preimage for t. By giving more possible choices for the decomposition of f, one can hope for better attacks. This however comes at the cost that they are now pseudo-preimage attacks, as one does not control the value of the IV anymore which is now equal to $t - f_3(s)$.

A possible improvement to a splice-and-cut decomposition is the use of *initial structures* [18], which were later reformulated as *bicliques* [10]. Instead of starting the computations in the middle from an intermediate value s, the idea is now to start from a set of multiple values possessing a special structure that spans several rounds. If the cost of constructing such sets is negligible w.r.t the rest of the computations, the rounds spanned by the structure actually come for free. In more details, a biclique, say for f_3 in the above decomposition of f, is a set $\{m, D_1, D_2, Q_1, Q_2\}$ where m is a message, D_1 and D_2 are linear spaces of dimension d, and Q_1 (resp. Q_2) is a list of 2^d values indexed by the differences δ_1 of D_1 (resp. D_2) s.t. $\forall(\delta_1, \delta_2) \in D_1 \times D_2 \quad Q_2[\delta_2] = f_3(m \oplus \delta_1 \oplus \delta_2, Q_1[\delta_1])$. This allows to search the message space $m \oplus D_1 \oplus D_2$ in $\mathcal{O}(2^d)$ with a meet-in-the-middle approach that does not need any call to f_3, essentially bypassing this part of the decomposition.

3 Higher-Order Differential Meet-in-The-Middle

We now describe how to modify the framework of §2 to use higher-order differentials. Let us denote by $(\{\alpha_1, \alpha_2\}, \{\beta_1, \beta_2\}) \xrightarrow[p]{f} \gamma$ the fact that $\Pr_{(x,y)}\big[f(x \oplus \alpha_1 \oplus \alpha_2, y \oplus \beta_1 \oplus \beta_2) \oplus f(x \oplus \alpha_1, y \oplus \beta_1) \oplus f(x \oplus \alpha_2, y \oplus \beta_2) = f(x, y) \oplus \gamma\big] = p$, meaning that $(\{\alpha_1, \alpha_2\}, \{\beta_1, \beta_2\})$ is a related-key order-2 differential for f that holds with probability p.

Similarly as in §2, the goal of the attacker is to find four linear subspaces $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2}$ of $\{0,1\}^m$ in direct sum (cf. equation (5)) such that:

$$D_{1,1} \bigoplus D_{1,2} \bigoplus D_{2,1} \bigoplus D_{2,2} \tag{5}$$

$$\forall \delta_{1,1}, \delta_{1,2} \in D_{1,1} \times D_{1,2} \; \exists \; \Delta_1 \in \{0,1\}^v \text{ s.t. } (\{\delta_{1,1}, \delta_{1,2}\}, \{0,0\}) \xrightarrow[1]{f_1} \Delta_1 \tag{6}$$

$$\forall \delta_{2,1}, \delta_{2,2} \in D_{2,1} \times D_{2,2} \; \exists \; \Delta_2 \in \{0,1\}^v \text{ s.t. } (\{\delta_{2,1}, \delta_{2,2}\}, \{0,0\}) \xrightarrow[1]{{f_2}^{-1}} \Delta_2. \tag{7}$$

Then $M \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}$ is a *preimage* of $c + p$ if and only if $f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, c) = {f_2}^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, p)$ which is equivalent by the equations (6) and (7) to the equality:

$$\begin{array}{lcl} f_1(\mu \oplus \delta_{1,1} \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus & & {f_2}^{-1}(\mu \oplus \delta_{2,1}, \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \\ f_1(\mu \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus & = & {f_2}^{-1}(\mu \oplus \delta_{2,2}, \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \\ f_1(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus \Delta_1 & & {f_2}^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \Delta_2. \end{array} \tag{8}$$

We denote by $d_{i,j}$ the dimension of the sub-space $D_{i,j}$ for $i, j = 1, 2$. Then for a set M of messages $\mu \in \{0,1\}^m$) one can define $\#M$ affine sub-sets $\mu_i \oplus D_{1,1} \oplus D_{1,2} \oplus$

$D_{2,1} \oplus D_{2,2}$ of dimension $d_{1,1} + d_{1,2} + d_{2,1} + d_{2,2}$ (since the sub-spaces $D_{i,j}$ are in direct sum by hypothesis), which can be tested for a preimage using (8). This can be done efficiently by a modification of Alg. 1 into the following Alg. 2.

Algorithm 2: Testing $\mu \oplus D_{1,1} \oplus D_{1,2} \oplus D_{2,1} \oplus D_{2,2}$ for a preimage

Input: $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0,1\}^m$, $\mu \in \{0,1\}^m$, p, c
Output: A preimage of $c + p$ if there is one in $\mu \oplus D_{1,1} \oplus D_{1,2}, \oplus D_{2,1} \oplus D_{2,2}$, $\perp$ otherwise
Data: Six lists:
$L_{1,1}$ indexed by $\delta_{1,2}, \delta_{2,1}, \delta_{2,2}$
$L_{1,2}$ indexed by $\delta_{1,1}, \delta_{2,1}, \delta_{2,2}$
$L_{2,1}$ indexed by $\delta_{1,1}, \delta_{1,2}, \delta_{2,2}$
$L_{2,2}$ indexed by $\delta_{1,1}, \delta_{1,2}, \delta_{2,1}$
L_1 indexed by $\delta_{2,2}, \delta_{2,1}$
L_2 indexed by $\delta_{1,1}, \delta_{1,2}$

1 **forall the** $\delta_{1,2}, \delta_{2,1}, \delta_{2,2} \in D_{1,2} \times D_{2,1} \times D_{2,2}$ **do**
2 $\quad L_{1,1}[\delta_{1,2}, \delta_{2,1}, \delta_{2,2}] \hookleftarrow {f_2}^{-1}(\mu \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, c)$;
3 **forall the** $\delta_{1,1}, \delta_{2,1}, \delta_{2,2} \in D_{1,1} \times D_{2,1} \times D_{2,2}$ **do**
4 $\quad L_{1,2}[\delta_{1,1}, \delta_{2,1}, \delta_{2,2}] \hookleftarrow {f_2}^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{2,1} \oplus \delta_{2,2}, c)$;
5 **forall the** $\delta_{1,1}, \delta_{1,2}, \delta_{2,2} \in D_{1,1} \times D_{1,2} \times D_{2,2} \times$ **do**
6 $\quad L_{2,1}[\delta_{1,1}, \delta_{1,2}, \delta_{2,2}] \hookleftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,2}, p)$;
7 **forall the** $\delta_{1,1}, \delta_{1,2}, \delta_{2,1} \in D_{1,1} \times D_{1,2} \times D_{2,1}$ **do**
8 $\quad L_{2,2}[\delta_{1,1}, \delta_{1,2}, \delta_{2,1}] \hookleftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1}, p)$;
9 **forall the** $\delta_{1,1}, \delta_{1,2} \in D_{1,1} \times D_{1,2}$ **do**
10 $\quad L_2[\delta_{1,1}, \delta_{1,2}] \hookleftarrow {f_2}^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \Delta_1$;
11 **forall the** $\delta_{2,1}, \delta_{2,2} \in D_{2,1} \times D_{2,2}$ **do**
12 $\quad L_1[\delta_{2,1}, \delta_{2,2}] \hookleftarrow f_1(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus \Delta_2$;
13 **forall the** $\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2} \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$ **do**
14 $\quad$ **if** $L_{1,1}[\delta_{1,2}, \delta_{2,1}, \delta_{2,2}] \oplus L_{1,2}[\delta_{1,1}, \delta_{2,1}, \delta_{2,2}] \oplus L_1[\delta_{2,1}, \delta_{2,2}] = L_{2,1}[\delta_{1,1}, \delta_{1,2}, \delta_{2,2}] \oplus L_{2,2}[\delta_{1,1}, \delta_{1,2}, \delta_{2,1}] \oplus L_2[\delta_{1,1}, \delta_{1,2}]$ **then**
15 $\quad\quad$ **return** $\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}$
16 **return** $\perp$

Analysis of Alg. 2. If we denote by Γ_1 and Γ_2 the cost of evaluating of f_1 and ${f_2}^{-1}$ and Γ_{match} the cost of the test on line 14, then the algorithm allows to test $2^{d_{1,1}+d_{1,2}+d_{2,1}+d_{2,2}}$ messages with a complexity of $2^{d_{1,2}+d_{2,1}+d_{2,2}}\Gamma_2 + 2^{d_{1,1}+d_{2,1}+d_{2,2}}\Gamma_2 + 2^{d_{1,1}+d_{1,2}+d_{2,1}}\Gamma_1 + 2^{d_{1,1}+d_{1,2}+d_{2,1}}\Gamma_1 + 2^{d_{1,1}+d_{1,2}}\Gamma_2 + 2^{d_{2,1}+d_{2,2}}\Gamma_1 + \Gamma_{\text{match}}$. The algorithm must then be run $2^{n-(d_{1,1}+d_{1,2}+d_{2,1}+d_{2,2})}$ times in order to test 2^n messages. In the special case where all the linear spaces have the same dimension d and if we consider that Γ_{match} is negligible with respect to the total complexity, the total complexity of an attack is then of : $2^{n-4d} \cdot (2^{3d} \cdot (2\Gamma_1 + 2\Gamma_2) + 2^{2d} \cdot (\Gamma_1 + \Gamma_2)) = 2^{n-d+1}\Gamma + 2^{n-2d}\Gamma = \mathcal{O}(2^{n-d})$ where Γ is the cost of the evaluation of the total compression function f. We think that the assumption on the cost of Γ_{match} to be reasonable given the small size of d in actual attacks and the fact that performing a single match is much faster than computing f.

The factor that is gained from a brute-force search of complexity $\mathcal{O}(2^n)$ is hence of 2^d, which is the same as for Alg. 1. However, one now needs four spaces of differences of size 2^d instead of only two, which might look like a setback. Indeed the real interest of this method does not lie in a simpler attack but in the fact that using higher-order

differentials may now allow to attack functions for which no good-enough order-1 differentials are available.

Using probabilistic truncated differentials. Similarly as in §2, Alg. 2 can be modified in order to use probabilistic truncated differentials instead of probability-1 differentials on the full state. The changes to the algorithm and the complexity evaluation are identical to the ones described in §2.2, which we refer to for a description.

4 Applications to SHA-1

4.1 Description of SHA-1

SHA-1 is an NSA-designed hash function standardized by the NIST [15]. It combines a compression function which is a block cipher with 512-bit keys and 160-bit messages used in Davies-Meyer mode with a Merkle-Damgård mode of operation [14, Chap. 9]. Thus, the initial vector (IV) as well as the final hash are 160-bit values, and messages are processed in 512-bit blocks. The underlying block cipher of the compression function can be described as follows: let us denote by $m_0, \dots m_{15}$ the 512-bit key as 16 32-bit words. The *expanded* key $w_0, \dots w_{79}$ is defined as:

$$w_i = \begin{cases} m_i & \text{if } i < 16 \\ (w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16}) \lll 1 & \text{otherwise.} \end{cases}$$

Then, if we denote by a, b, c, d, e a 160-bit state made of 5 32-bit words and initialized with the plaintext, the ciphertext is the value held in this state after iterating the following procedure (parametered by the round number i) 80 times:

$$\begin{aligned} t &\leftarrow\!\shortmid (a \lll 5) + \Phi_{i \div 20}(b, c, d) + e + k_{i \div 20} + w_i \\ e &\leftarrow\!\shortmid d \\ d &\leftarrow\!\shortmid c \\ c &\leftarrow\!\shortmid b \lll 30 \\ b &\leftarrow\!\shortmid a \\ a &\leftarrow\!\shortmid t, \end{aligned}$$

where '$\div$' denotes the integer division, $\Phi_{0\dots3}$ are four bitwise Boolean functions, and $k_{0\dots3}$ are four constants (we refer to [15] for their definition).

Importantly, before being hashed, a message is *always* padded with at least 65 bits, made of a '1' bit, a (possibly zero) number of '0' bits, and the length of the message in bits as a 64-bit integer. This padding places an additional constraint on the attacker as it means that even a preimage for the compression function with a valid IV is not necessarily a preimage for the hash function.

4.2 One-block preimages without padding

We apply the framework of §3 to mount attacks on SHA-1 for *one-block preimages without padding*. These are rather direct applications of the framework, the only difference being the fact that we use *sets* of differentials instead of *linear spaces*. This has no impact on Alg. 2, but makes the description of the attack parameters less compact.

As was noted in [11], the message expansion of SHA-1 being linear, it is possible to attack 15 steps both in the forward and backward direction (for a total of 30)

without advanced matching techniques: it is sufficient to use a message difference in the kernel of the 15 first steps of the message expansion. When applying our framework to attack more steps (say 55 to 62), we have observed experimentally that splitting the forward and backward parts around steps 22 to 27 seems to give the best results. A similar behaviour was observed by Knellwolf and Khovratovich in their attacks, and this can be explained by the fact that the SHA-1 step function has a somewhat weaker diffusion when computed backward compared to forward.

We use Alg. 3 to construct a suitable set of differences in the preparation of an attack. This algorithm was run on input differences of low Hamming weight; these are kept only when they result in output differences with truncation masks that are long enough and with good overall probabilities. The sampling parameter Q that we used was 2^{15}; the threshold value t was subjected to a tradeoff: the larger it is, the less bits are chosen in the truncation mask, but the better the probability of the resulting differential. In practice, we used values between 2 and 5, depending on the differential considered.

Algorithm 3: Computing a suitable output difference for a given input difference

Input: A chunk f_i of the compression function, $\delta_{i,1}, \delta_{i,2} \in \{0,1\}^m$, a threshold value t, a sample size Q, an internal state c.

Output: An output difference $\mathcal{S}$, and a mask $T_{\mathcal{S}}$ for the differential $((\delta_{i,1}, \delta_{i,2}), 0) \overset{f_i}{\rightsquigarrow} \mathcal{S}$

Data: An array d of n counters initially set to 0.

```
for q = 0 to Q do
    Choose μ ∈ {0,1}^m at random ;
    Δ ← f_i(μ ⊕ δ_{i,1} ⊕ δ_{i,2}, c) ⊕ f_i(μ ⊕ δ_{i,1}, c) ⊕ f_i(μ ⊕ δ_{i,2}, c) ⊕ f_i(μ, c);
    for i = 0 to n − 1 do
        if the i^th bit of Δ is 1 then
            d[i] ← d[i] + 1;
for i = 0 to n − 1 do
    if d[i] ≥ t then
        Set the i-th bit of the output difference S to 1;
```

Once input and output differences have been chosen, we use an adapted version of Alg. 2 from [11] given in Alg. 4 to compute suitable truncation masks.

The choice of the size of the truncation mask d in this algorithm offers a tradeoff between the probability one can hope to achieve for the resulting truncated differential and how efficient a filtering of "bad" messages it will offer. In our applications to SHA-1, we chose masks of size about $\min(\log_2(|D_{1,1}|), \log_2(|D_{1,2}|), \log_2(|D_{2,1}|), \log_2(|D_{2,2}|))$, which is consistent with taking masks of size the dimension of the affine spaces as is done in [11].

We similarly adapt Alg. 3 from [11] as Alg. 5 in order to estimate the average false negative probability associated with the final truncated differential.

We conclude this section by giving the statistics for the best attacks that we found for various reduced versions of SHA-1 in Table 2, the highest number of attacked rounds being 62. Because the difference spaces are no longer affine, they do not lend themselves to a compact description and their size is not necessarily a power of 2 anymore. The ones we use do not have many elements, however, which makes them easy to enumerate.

Algorithm 4: Find truncation mask T for matching

Input: $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0,1\}^m$, a sample size Q, a mask size d.
Output: A truncation mask $T \in \{0,1\}^n$ of Hamming weight d.
Data: An array k of n counters initially set to 0.

for $q = 0$ *to* Q **do**
  Choose $\mu \in \{0,1\}^m$ at random ;
  $c \leftarrow\!\!\rightarrow f(\mu, iv)$;
  Choose $(\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2}) \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$ at random;
  $\Delta \leftarrow\!\!\rightarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus f_1(\mu \oplus \delta_{1,1}, c) \oplus f_1(\mu \oplus \delta_{1,2}, c)$;
  $\Delta \leftarrow\!\!\rightarrow \Delta \oplus {f_2}^{-1}(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, c) \oplus {f_2}^{-1}(\mu \oplus \delta_{2,2}, c) \oplus {f_2}^{-1}(\mu \oplus \delta_{2,2}, c)$;
  for $i = 0$ *to* $n - 1$ **do**
    if *the* i^{th} *bit of* Δ *is 1* **then**
      $k[i] \leftarrow\!\!\rightarrow k[i] + 1$;
Set the d bits of lowest counter value in k to 1 in T.

Algorithm 5: Estimate the average false negative probability

Input: $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0,1\}^m, T \in \{0,1\}^n$, a sample size Q
Output: Average false negative probability α.
Data: A counter k initially set to 0.

for $q = 0$ *to* Q **do**
  Choose $\mu \in \{0,1\}^m$ at random ;
  $c \leftarrow\!\!\rightarrow f(\mu, iv)$;
  Choose $(\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2}) \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$ at random;
  $\Delta \leftarrow\!\!\rightarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus f_1(\mu \oplus \delta_{1,1}, c) \oplus f_1(\mu \oplus \delta_{1,2}, c)$;
  $\Delta \leftarrow\!\!\rightarrow \Delta \oplus {f_2}^{-1}(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, c) \oplus {f_2}^{-1}(\mu \oplus \delta_{2,2}, c) \oplus {f_2}^{-1}(\mu \oplus \delta_{2,2}, c)$;
  for $i = 0$ *to* $n - 1$ **do**
    if $\Delta \not\equiv_T 0^n$ **then**
      $k \leftarrow\!\!\rightarrow k + 1$;
return k/Q

4.3 One-block preimages with padding

If we want the message to be properly padded, 65 out of the 512 bits of the last message blocks need to be fixed according to the padding rules, and this naturally restricts the positions of where one can now use message differences. This has in particular an adverse effect on the differences in the backward step, which Hamming weight increases because of some features of SHA-1's message expansion algorithm. The overall process of finding attack parameters is otherwise unchanged from the non-padded case. We give statistics for the best attacks that we found in Table 3. One will note that the highest number of attacked rounds dropped from 62 to 56 when compared to Table 2.

4.4 Two-block preimages with padding

We can increase the number of rounds for which we can find a preimage with a properly padded message at the cost of using a slightly longer message of two blocks: if we are able to find *one-block pseudo preimages with padding* on enough rounds, we can then use the *one-block preimage without padding* to bridge the former to the IV. Indeed, in a pseudo-preimage setting, the additional freedom degrees gained from removing any constraint on the IV more than compensate for the ones added by the padding. We first describe how to compute such pseudo-preimages.

Table 2. One block preimage without padding. N is the number of attacked steps, *Split* is the separation step between the forward and the backward chunk, $d_{i,j}$ is the $\log_2$ of the cardinal of $D_{i,j}$ and α is the estimate for the false negative probability. The complexity is computed as described in §3.

N	*Split*	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	α	Complexity
58	25	7.6	9.0	9.2	9.0	0.73	157.4
59	25	7.6	9.0	6.7	6.7	0.69	157.7
60	26	6.5	6.0	6.7	6.0	0.60	158.0
61	27	4.7	4.8	5.7	5.8	0.51	158.5
62	27	4.7	4.8	4.3	4.6	0.63	159.0

Table 3. One block preimage with padding. N is the number of attacked steps, *Split* is the separation step between the forward and the backward chunk, $d_{i,j}$ is the $\log_2$ of the cardinal of $D_{i,j}$ and α is the estimation for false negative probability. The complexity is computed as described in §3.

N	*Split*	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	α	Complexity
51	23	8.7	8.7	8.7	8.7	0.72	155.6
52	23	9.1	9.1	8.2	8.2	0.61	156.7
53	23	9.1	9.1	3.5	3.5	0.61	157.7
55	25	6.5	6.5	5.9	5.7	0.52	158.2
56	25	6	6.2	7.2	7.2	0.6	159.4

One-block pseudo-preimages. If we relax the conditions on the IV and do not impose anymore that it is fixed to the one of the specifications, it becomes possible to use a splice-and-cut decomposition of the function, as well as short (properly padded) bicliques.

The (reduced) compression function of SHA-1 f is now decomposed into three smaller functions as $f = {f_2}^t \circ f_1 \circ f_3 \circ {f_2}^b$, f_3 being the rounds covered by the biclique. The function f_1 covers the steps s_1 to e, $f_2 = {f_2}^t \circ {f_2}^b$ covers s_2 to $e+1$ through the feedforward, and f_3 covers s_2+1 to s_1-1, as shown in Figure 1.

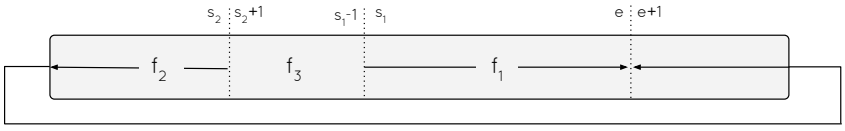

Fig. 1. A splice-and-cut decomposition with biclique.

Finding the parameters is done in the exact same way as for the one-block preimage attacks. Since the bicliques only cover 7 steps, one can generate many of them from a single one by modifying some of the remaining message words outside of the biclique proper. Therefore, the amortized cost of their construction is small and considered negligible w.r.t. the rest of the attack. The resulting attacks are shown in Table. 4.

Table 4. One block pseudo-preimage with padding. N is the number of attacked steps, $d_{i,j}$ is the $\log_2$ of the cardinal of the set $D_{1,2}$ and α is the estimation for false negative probability. The various splits are labeled as in Figure 1. The complexity is computed as described in §3.

N	s_1	e	s_2	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	α	Complexity
61	27	49	20	7.0	7.0	7.5	7.5	0.56	156.4
62	27	50	20	5.8	5.7	7.2	7.2	0.57	157.0
63	27	50	20	6.7	6.7	7.7	7.7	0.57	157.6
64	27	50	20	3	3	4.5	4.7	0.69	158.7

Complexity of the two-block attacks. Using both one-block attacks, it is simple to mount a two-block attack at the combined cost of each of them. For a given target c, one:

1. uses a properly-padded pseudo-preimage attack, yielding the second message block μ_2 and an IV iv';
2. uses a non-padded preimage attack with target iv', yielding a first message block μ_2.

From the Merkle-Damgård structure of SHA-1, it follows that the two-block message (μ_1, μ_2) is a preimage of c.

For attacks up to 60 rounds, we can use the pseudo-preimage attacks of [11]; for 61 and 62 rounds, we use the ones of this section. This results in attacks of complexities as shown in Table 5.

Table 5. Two-block preimage attacks on SHA-1 reduced to N steps. The pseudo-preimage attacks followed by '$\star$' come from [11].

N	Second block complexity	First block complexity	Total Complexity
58	$156.3^\star$	157.4	157.9
59	$156.7^\star$	157.7	158.3
60	$157.5^\star$	158.0	158.7
61	156.4	158.5	158.8
62	157.0	159.0	159.3

5 Applications to BLAKE and BLAKE2

5.1 Description of BLAKE

The hash function BLAKE [3] was a candidate and one of the five finalists of the SHA-3 competition, that ended in November 2012. Although it was not selected as the winner, no weaknesses were found in BLAKE and it is accepted as being a very secure and efficient hash function [6]. More recently, a faster variant BLAKE2 has been designed [5]; both functions come in two variants with a chaining value of 256 bits (BLAKE-256 and BLAKE2s) and 512 bits (BLAKE-512 and BLAKE2b). The design of BLAKE is somewhat similar to the one of SHA-1, as being built around a compression function in Merkle-Damgård mode. It does however feature a few notable differences: first, the compression function takes two additional inputs to the

message m and the previous chaining value c, in the form of a user-defined salt s and a block counter t. The new chaining value c' is thus defined as $c' \triangleq h(c, m, s, t)$; second, the compression function follows the local wide-pipe paradigm which was introduced by BLAKE's predecessor LAKE [4], meaning that the state size of the compression function h is larger than the size of the chaining value c. In particular, this implies that c is first expanded to form the input to h, and that the output of the latter is compressed in order to give c'. This feature has some important consequences when analyzing the function and makes some types of attacks harder to perform, as we will see later. We describe BLAKE-512 in more details, and refer to the specification document [3] for a full description of the function and of its variants. Similarly, the changes from BLAKE to BLAKE2 having no impact on our overall attack strategy, we refer the reader to the specifications of BLAKE2 for more details [5].

Initialization and finalization of the compression function. The state of the compression function is logically seen as a 4×4 array of 64-bit words $v_{0\ldots15}$, making 1024 bits in total. It is initialized from 8 words of incoming chaining value $c_{0\ldots7}$, 4 words of salt $s_{0\ldots4}$, 2 words of counter $t_{0,1}$ and 8 words of constant $k_{0\ldots7}$, as shown below:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow\!\shortmid \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ s_0 \oplus k_0 & s_1 \oplus k_1 & s_2 \oplus k_2 & s_3 \oplus k_3 \\ t_0 \oplus k_4 & t_0 \oplus k_5 & t_1 \oplus k_6 & t_1 \oplus k_7 \end{pmatrix}.$$

The outgoing chaining value $c'_{0\ldots7}$ is defined from the final value of the state $v'_{0\ldots15}$, the initial value of the chaining value and the salt as:

$$\begin{array}{ll} c'_0 \leftarrow\!\shortmid c_0 \oplus s_0 \oplus v'_0 \oplus v'_8 & c'_4 \leftarrow\!\shortmid c_4 \oplus s_0 \oplus v'_4 \oplus v'_{12} \\ c'_1 \leftarrow\!\shortmid c_1 \oplus s_1 \oplus v'_1 \oplus v'_9 & c'_5 \leftarrow\!\shortmid c_5 \oplus s_1 \oplus v'_5 \oplus v'_{13} \\ c'_2 \leftarrow\!\shortmid c_2 \oplus s_2 \oplus v'_2 \oplus v'_{10} & c'_6 \leftarrow\!\shortmid c_6 \oplus s_2 \oplus v'_6 \oplus v'_{14} \\ c'_3 \leftarrow\!\shortmid c_3 \oplus s_3 \oplus v'_3 \oplus v'_{11} & c'_7 \leftarrow\!\shortmid c_7 \oplus s_3 \oplus v'_7 \oplus v'_{15} \end{array}.$$

Round function. One round of BLAKE is made of eight calls to a 'quarter-round' function G_i on part of the state:

$$\begin{array}{llll} \mathrm{G}_0(v_0, v_4, v_8, v_{12}) & \mathrm{G}_1(v_1, v_5, v_9, v_{13}) & \mathrm{G}_2(v_2, v_6, v_{10}, v_{14}) & \mathrm{G}_3(v_3, v_7, v_{11}, v_{15}) \\ \mathrm{G}_4(v_0, v_5, v_{10}, v_{15}) & \mathrm{G}_5(v_1, v_6, v_{11}, v_{12}) & \mathrm{G}_6(v_2, v_7, v_8, v_{13}) & \mathrm{G}_7(v_3, v_4, v_9, v_{14}) \end{array}.$$

There are 16 such rounds for BLAKE-512[2]. Furthermore, because the whole state is updated twice during one round (once by $\mathrm{G}_{0\ldots3}$ and once by $\mathrm{G}_{4\ldots7}$), one such update will be called a half-round. The function $\mathrm{G}_i(a, b, c, d)$ is defined for round r as:

$$\begin{array}{ll} 1 : a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus k_{\sigma_r(2i+1)}) & 5 : a \leftarrow a + b + (m_{\sigma_r(2i+1)} + k_{\sigma_r(2i)}) \\ 2 : d \leftarrow (d \oplus a) \ggg 32 & 6 : d \leftarrow (d \oplus a) \ggg 16 \\ 3 : c \leftarrow c + d & 7 : c \leftarrow c + d \\ 4 : b \leftarrow (b \oplus c) \ggg 25 & 8 : b \leftarrow (b \oplus c) \ggg 11 \end{array},$$

with σ a round-dependent permutation of $\{0 \ldots 15\}$. The padding is nearly the same as for SHA-1. The only difference is that a '1' bit is again systematically appended after the '0' bits (if any). Hence, there are at least 66 bits of padding.

[2] There are 14 for BLAKE-256, 12 for BLAKE2b and 10 for BLAKE2s.

Terminology. As can be seen from the above description, there is an additional initialization phase for the compression function in BLAKE when compared to most hash functions and SHA-1 in particular. We choose to call *pseudo-preimage on the compression function* a preimage attack that bypasses this initialization and requires complete freedom for the initial 16-word state, yet that does respect the finalization (and thence forms an attack for the same level of security than the compression function, *i.e.* 512 bits for BLAKE-512). This is a more restrictive model than attacking the underlying block cipher of the compression function in a PGV mode, which would in itself be a significant attack on a building block of the hash function.

5.2 Pseudo-preimage on the compression function

If we relax the conditions on the initialization of the compression function, we can use a splice-and-cut approach to mount a pseudo-preimage attack in a straight application of the framework, which we did on the BLAKE(2) family. We note that because of the use of a local-wide-pipe construction, matching in the middle of the compression function has a complexity the square of the actual security level for preimages. Therefore we perform the matching phase right on the output of the compression function.

We mount an attack on the compression function reduced to 7.5 rounds of BLAKE-512 and BLAKE 2b attacking round 0.5 (resp. 0.25) to round 8 (resp. 7.75), and 6.75 rounds of BLAKE-256 and BLAKE2s attacking round 0.75 to round 7.5. We decompose this reduced compression function f as $f_1 \circ f_3 \circ f_2$. The function f_1 starts at round 5.25 and ends at round 8 (resp. 7.75 for BLAKE2b and 7.5 for BLAKE2s and BLAKE-256), ${f_2}^{-1}$ covers rounds 4 to 0.5 (resp. 0.25 and 0.75) , and f_3 is a biclique covering the 0.75 remaining rounds. Finding the parameters of the attack is done similarly as in §4. Since the biclique covers less than one round, it leaves some of the message words free, which can then be used to generate many similar bicliques. Therefore, the amortized cost of their construction is small and considered negligible w.r.t. the rest of the attack. The only message words with differences are m_2 & m_8 in the forward computation and m_3 & m_{11} in the backward computation. As a consequence, the whole message can easily be properly padded.The statistics of the resulting attacks are shown in Table. 6.

Table 6. One block pseudo-preimage without padding on BLAKE-512 and BLAKE2b; $d_{i,j}$ is the $\log_2$ of the cardinal of the set $D_{i,j}$ and α is the estimation for false negative probability. The complexity is computed as described in §3. The various splits are labeled as in Figure 1.

Function	Start	s_1	e	s_2	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	α	Complexity
BLAKE-512	0.5	5.25	8	4	3.0	9.3	4.0	9.5	0.49	510.3
BLAKE2b	0.25	5.25	7.75	4	4.5	8.0	3.9	3.9	0.41	510.9
BLAKE-256	0.75	5.25	7	4	4.1	7.2	9.0	9.0	0.64	253.9
BLAKE2s	0.75	5.25	7	4	4.1	7.2	9.0	9.0	0.68	253.8

5.3 Preimage on the hash function

We now adapt the framework to mount a preimage attack for the larger variants of BLAKE and BLAKE2. Because of the quick diffusion of BLAKE's round func-

tion (notably due to the fact that every message word is used in every round), we were unsuccessful when searching for difference spaces resulting in a good meet-in-the-middle decomposition that simultaneously preserves the initialization of the compression function.

To overcome this problem, we use a single difference space, in the forward direction only. The use of order-2 differentials proves to be critical at this point, as no gain could be easily obtained otherwise. More precisely, we use differences of the type $(\{\alpha_1, \alpha_2\}, \{0, 0\}) \overset{f}{\underset{p}{\rightsquigarrow}}_T 0$, meaning that with probability p over the messages m, $f(m) \oplus f(m \oplus \alpha_1) \oplus f(m \oplus \alpha_2) \equiv f(m \oplus \alpha_1 \oplus \alpha_2)$ for a truncation mask T. This equality can be used with Alg. 2 modified as Alg. 6 in an attack. The basic idea at play here is that after computing $f(m)$, $f(m \oplus \alpha_1)$ and $f(m \oplus \alpha_2)$, one can test the values of $f(m \oplus \alpha_1 \oplus \alpha_2)$ for essentially no additional cost. We give an illustration of this process in Figure 2.

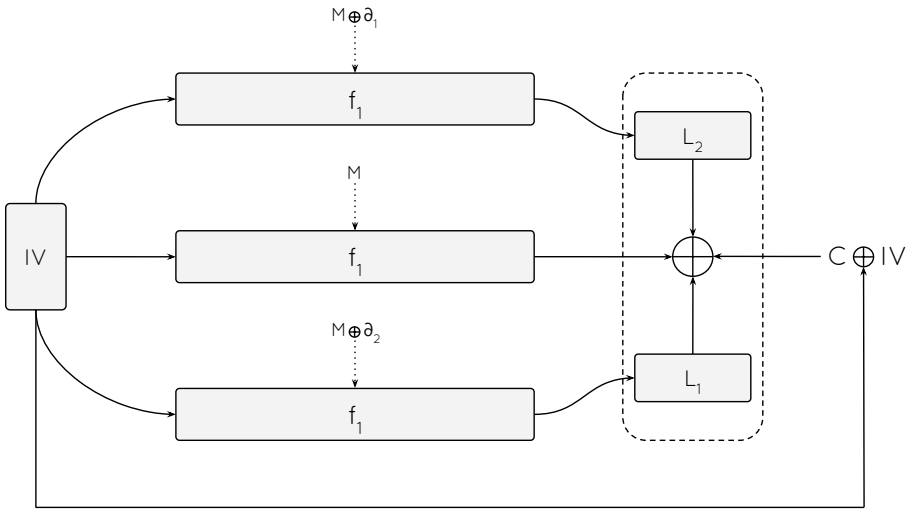

Fig. 2. Speeding-up preimage search with one-way order-2 differentials.

Algorithm 6: Testing $\mu \oplus D_{1,1} \oplus D_{1,2}$ for a preimage

Input: $D_{1,1}, D_{1,2} \subset \{0,1\}^m$, $\mu \in \{0,1\}^m$, p, c
Output: A preimage if there is one in $\mu \oplus D_{1,1} \oplus D_{1,2}$, $\perp$ otherwise
Data: Two lists L_1, L_2 indexed by $\delta_{1,2}, \delta_{1,1}$ respectively

1 **forall the** $\delta_{1,2} \in D_{1,2}$ **do**
2 $L_1[\delta_{1,2}] \hookleftarrow f_1(\mu \oplus \delta_{1,2}, p)$
3 **forall the** $\delta_{1,1} \in D_{1,1}$ **do**
4 $L_2[\delta_{1,1}] \hookleftarrow f_1(\mu \oplus \delta_{1,1}, c)$
5 **forall the** $(\delta_{1,1}, \delta_{1,2}) \in D_{1,1} \times D_{1,2}$ **do**
6 **if** $L_1[\delta_{1,2}] \oplus L_2[\delta_{1,1}] \equiv f_1(\mu, c) \oplus c \oplus p$ **then**
7 **return** $\mu \oplus \delta_{1,1} \oplus \delta_{1,2}$
8 **return** $\perp$

Analysis of Alg. 6. The test on line 6 can be performed efficiently by using an appropriate data structure (typically a hash table), resulting in overall linear time for the loop on line 5. In line with §2.2, the total complexity of an attack based on this algorithm thus becomes $(2^{n-d_1-d_2}(2^d_1 + 2^d_2)\Gamma + s)/\tilde{p}$, where d_1 (resp. d_2) is the dimension of $D_{1,1}$ (resp. $D_{1,2}$), Γ is the cost of calling f_1, s the complexity of retesting and $\tilde{p}$ is the average success probability of the order-2 differentials.

Application to BLAKE(2). We introduce the differences at round 5.25 and let them propagate with good probability for 2.75 rounds for BLAKE-512 and BLAKE2b. Like in §5.3 the only message words with differences are m_2 & m_8 in the forward computation and m_3 & m_{11} in the backward computation. As a consequence, the whole message can easily be properly padded.The attack parameters are found as before and lead to the attacks in Table 7.

Table 7. The preimage attacks on BLAKE(2); $d_{i,j}$ is the $\log_2$ of the cardinal of $D_{i,j}$ and α is the estimate for the false negative probability.

Function	#rounds	$d_{1,1}$	$d_{1,2}$	α	Complexity
BLAKE-512	2.75	4.0	9.5	0.6	510.3
BLAKE2b	2.75	3.1	6.9	0.4	511.0

References

1. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5381, pp. 103–119. Springer (2008), http://dx.doi.org/10.1007/978-3-642-04159-4_7
2. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) Advances in Cryptology — CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 70–89. Springer (2009), http://dx.doi.org/10.1007/978-3-642-03356-8_5
3. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE, version 1.3 (2008), available online at https://131002.net/blake/
4. Aumasson, J., Meier, W., Phan, R.C.: The Hash Function Family LAKE. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 36–53. Springer (2008), http://dx.doi.org/10.1007/978-3-540-71039-4_3
5. Aumasson, J., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: Simpler, Smaller, Fast as MD5. In: Jr., M.J.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7954, pp. 119–135. Springer (2013), http://dx.doi.org/10.1007/978-3-642-38980-1_8
6. Chang, S.j., Perlner, R., Burr, W.E., Turan, M.S., Kelsey, J.M., Paul, S., Bassham, L.E.: Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. NIST Interagency Report 7896 (2012)
7. Diffie, W., Hellman, M.E.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10, 74–84 (1977)
8. Guo, J., Karpman, P., Nikolic, I., Wang, L., Wu, S.: Analysis of BLAKE2. In: Benaloh, J. (ed.) Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings. Lecture Notes

in Computer Science, vol. 8366, pp. 402–423. Springer (2014), http://dx.doi.org/10.1007/978-3-319-04852-9_21
9. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 56–75. Springer (2010)
10. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 244–263. Springer (2012), http://dx.doi.org/10.1007/978-3-642-34047-5_15
11. Knellwolf, S., Khovratovich, D.: New Preimage Attacks against Reduced SHA-1. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology — CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 367–383. Springer (2012), http://dx.doi.org/10.1007/978-3-642-32009-5_22
12. Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. In: Communications and Cryptography, pp. 227–233. Springer (1994)
13. Li, J., Xu, L.: Attacks on Round-Reduced BLAKE. IACR Cryptology ePrint Archive 2009, 238 (2009), http://eprint.iacr.org/2009/238
14. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
15. National Institute of Standards and Technology: FIPS 180-4: Secure Hash Standard (SHS) (March 2012)
16. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) Advances in Cryptology — CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings. Lecture Notes in Computer Science, vol. 773, pp. 368–378. Springer (1993), http://dx.doi.org/10.1007/3-540-48329-2_31
17. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology — CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 303–311. Springer (1990), http://dx.doi.org/10.1007/3-540-38424-3_22
18. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) Advances in Cryptology — EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 134–152. Springer (2009), http://dx.doi.org/10.1007/978-3-642-01001-9_8
19. Wang, L., Ohta, K., Sakiyama, K.: Free-start preimages of round-reduced BLAKE compression function. ASIACRYPT rump session (2009), http://www.iacr.org/conferences/asiacrypt2009//rump/slides/11.pdf
20. Zhong, J., Lai, X.: Improved preimage attack on one-block MD4. Journal of Systems and Software 85(4), 981–994 (2012)