Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family

Dmitry Khovratovich[1] and Christian Rechberger[2] and Alexandra Savelieva[1]

[1]Microsoft Research Redmond
[2]ENS Paris and Chaire France Telecom

Abstract

We introduce a new concept in splice-and-cut attacks on hash functions, which bridges the gap between preimage attacks and a powerful method of differential cryptanalysis. The new concept is called biclique, for its system of equations resembling a complete bipartite graph. In view of the current SHA-3 competition, we apply our method to the finalist Skein and demonstrate the first attack on a 22-round version of Skein-512 in the most relevant hash function setting. Then we present the best attacks on the SHA-2 family of hash functions, breaking 45 out of the 64 rounds of SHA-256, 50 rounds of the 80 rounds of SHA-512, and many more rounds in the less relevant compression function setting.

Keywords: SHA-2, SHA-256, SHA-512, Skein, SHA-3, hash function, meet-in-the-middle attack, splice-and-cut, preimage attack, initial structure, biclique.

1 Introduction

The last years saw a dramatic progress in preimage attacks on hash functions. However, it was not always like this. While collision attacks progressed and essentially discarded MD5 [28, 25], preimage search seemed to be highly difficult. To a large extent this can be attributed to the concept of differential cryptanalyis [7], a powerful tool from the block cipher analysis, that naturally fits the idea of collision search. For preimage problem this technique did not appear useful since no object was identified where a difference would apply. An attempt to mount a second preimage attack in differential framework resulted in a dramatically low success rate [29].

The year 2008 brought preimage attacks on MD4 [17], MD5 [2, 23], Haval [22] and reduced SHA-0/1 [8]. While the methods of Leurent [17] and De Cannière-Rechberger [8] are rather dedicated, the idea of Aoki-Sasaki [2] proved to be widely applicable. They proposed to splice the first and the last rounds of the compression function based on the Davies-Meyer mode, thus making it computable in any direction. This resulted in the preimage attacks on full MD5 [24], and later the best attacks on the reduced SHA-0/1/2 hash functions [1, 3], and full Tiger [11].

Splice-and-cut framework and its progress. Both splice-and-cut and meet-in-the-middle attacks exploit the property that a part of a primitive does not make use of particular key/message bits. If the property holds, the computation of this part stands still if we flip those bits in the other part of a primitive. Assume the property is mutual, i.e. such bits can be found for both parts (also called *chunks*). Then a cryptanalyst prepares a set of independent computations for all possible values of those bits (called *neutral bits*) and subsequently checks for the match in the middle. The gain of the attack is proportional to the number of neutral bits.

Sasaki and Aoki observed [2, 23] that compression functions with permutation-based message schedule are extremely vulnerable to this kind of attack as chunks can be dramatically long. They also proposed various improvements. For example, since the number of computations to match decreases together with the number of neutral bits, the match can be performed on a small part

of the state. In turn, the matching bits depend on fewer message bits, which in fact leads to even larger number of neutral bits and the reduction in complexity.

The most interesting trick, however, is a so called initial structure [24, 3]. The initial structure can be informally defined as an overlapping of chunks, where neutral bits, although formally belonging to both chunks, are involved in computation of the proper chunk only. Concrete examples of the initial structure are much more sophisticated and hard to generalize. The concept seems to have large potential and few boundaries, while the other improvements are likely exhausted already. As a motivating example, consider the case of MD4, where recently an initial structure for as many as 17 rounds of a compression function was built. If this would work for SHA-256 (in the current best published attack this is limited to four rounds), we would just be a few rounds away from a full preimage attack on the hash standard.

Our contributions. We replace the idea of the initial structure with a more formal and general concept of biclique, which provides us with several layers of understanding and applications. We derive a system of functional equations linking internal states several rounds apart. Then we show that it is equivalent to a system of differentials, so the full structure of states can be built out of a structure of trails. These structures are two sets of internal states with each state having a relation with all states in another set. In terms of graph theory, these structures are referred to as *bicliques*. A differential view, that builds up on this formalism, allows us to apply numerous tools from collision search and enhanced differential attacks, from message modifications to local collisions. We propose several algorithms constructing these bicliques, which are generic and flexible.

The applications of our concept are broad. Our first and simple example is the hash function and the SHA-3 finalist Skein-512, which lacks any attacks in the hash setting. We develop an attack on 22 rounds of Skein-512, which is comparable to the best attacks on the compression function that survived the last tweak. Our attack on the compression function of Skein-512 utilizes many more degrees of freedom as we control the full input, and thus results in a 37-round attack.

Our second group of applications is the SHA-2 family. Enhanced with the differential analysis, we heavily use differential trails in SHA-2, message modification techniques from SHA-1 and SHA-0, and trail backtracking techniques from RadioGatun, Grindahl, SHA-1, and many others. As a result, we build attacks on 45-round SHA-256 and 50-round SHA-512, both the best attacks in the hash mode. Regarding the compression functions, we penetrate up to seven more rounds, thus reaching 52 rounds and violating the security of about 80% of SHA-256.

Reference	Target	Steps	Complexity		Memory (words)
			Pseudo-preimage	Preimage	
[1, 11]	SHA-256	43	$2^{251.9}$	$2^{254.9}$	2^6
Section 5.2	SHA-256	45	2^{253}	$2^{255.5}$	2^6
Section 6	SHA-256	52	2^{255}	-	2^6
[1, 11]	SHA-512	46	2^{509}	$2^{511.5}$	2^6
Section 5.3	SHA-512	50	2^{509}	$2^{511.5}$	2^4
Section 6	SHA-512	57	2^{511}	-	2^6
Section 5.3	Skein-512	22	2^{508}	2^{511}	2^6
Section 6	Skein-512	37	$2^{511.2}$	-	2^{64}

Table 1: Comparison of preimage attacks on the SHA-2 family and Skein-512.

2 Basics of splice-and-cut preimage attacks

In this section we describe the basic idea of splice-and-cut preimage attacks on a blockcipher-based compression function. We consider the most popular Davies-Meyer mode: $H = f(M, CV) =$

$E_M(CV) \oplus CV$, where CV is the chaining variable, and E is the block cipher keyed with M. We split the cipher E and message M into three parts: $E = E^3 \circ E^2 \circ E^1$ and $M = M^F||M^B||M^C$ such that

- M^F is not used in E_1 and E_3;
- M^B is not used in E_2.

Then we get the following chain of computations:

$$CV \xrightarrow[E^1]{M^B||M^C} S \xrightarrow[E^2]{M^F||M^C} V \xrightarrow[E^3]{M^B||M^C} H,$$

where S and V are the internal states. The basic attack works as follows.

1. Assign arbitrary values to S and M^C.
2. Compute states $\mathcal{V} = \{V \xleftarrow[E^2]{M^F} S\}$ for different M^F and store them in memory.
3. Compute states $\mathcal{V}' = \{V \xleftarrow[(E^1 \circ E^3)^{-1}]{M^F} S$ for different M^B and store them in memory. First, compute CV, then use CV and H to find the output of E^3, and then compute in the backward direction to V.
4. The set $\mathcal{V} \cap \mathcal{V}'$ constitutes preimages for the compression function. If necessary, repeat the procedure for other S and M^C.

The computations in Step 2 and Step 3 are called *forward* and *backward chunks*, and the elements of M^F and M^B are called *neutral bits*. Neutral bits might be bits of round messages and even linear spaces of bits. In the simple case when M^F and M^B have equal length d, the sets $\mathcal{V}$ and $\mathcal{V}'$ have 2^d elements, so the probability of a match is 2^{2d-n}, where n is the state size. Assuming that each chunk costs about $1/2$ calls of the compression function, the complexity to find a preimage is about $2^{n-2d+d} = 2^{n-d}$. The complexity grows negligibly, if we match not on the full state, but only on a subset of bits of size about d (partial matching [2], see also [9]).

To mount an attack on a hash function, one generates $2^{d/2}$ preimages of the compression function, and $2^{n-d/2}$ images of the original IV with different messages. Then the probability that there is a match in CV is high, and the complexity is $2^{n-d/2+1}$.

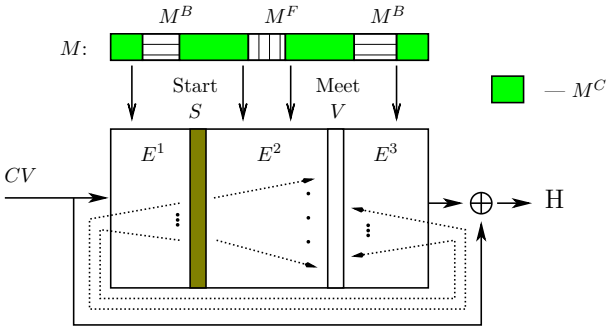

Figure 1: Basic splice-and-cut attack

Initial structure. Recently, cryptanalysts have replaced the starting state S with a sophisticated construction called the initial structure. The broad idea is to put constraints on internal variables and form S out of variables spread over several rounds. The matching procedure is the same, but the cryptanalysts have to prove separately that the construction is well-defined, i.e. that a match in the middle implies a correct computation in the start. The attacks enhanced with the initial structure cover two or 4 more rounds in SHA-2 [1, 11], 4 more rounds in SHA-1 [3] and 17 more rounds in MD4 [11]. Due to the lack of formalism introduced in those papers, we are unable to provide a short but meaningful overview of the constructions used there. Instead, we build up our own concept.

3 Bicliques and differential framework for splice-and-cut preimage attacks

3.1 Bicliques

We propose to replace an informal and heuristic construction of the initial structure with a formal concept. Our crucial observation is that at Steps 2-3 of the attack a computation of a chunk is always associated with a particular value of neutral bits, either M^B or M^F. Let the backward chunk start at a state Q, and denote by Q_i the state being computed with $M^B = i$. Let also P be the starting state for the forward chunk, and denote by P_j the state being computed with $M^F = j$. If matched at the point V, the combination (Q_i, P_j) must provide a preimage. Therefore, all the Q_i and P_j must follow the following conditions:

$$\forall i, j: \quad Q_i \xrightarrow[\mathcal{B}]{M_i^B || M_j^F} P_j, \tag{1}$$

where $\mathcal{B}$ transforms Q to P, and $M_i^B = i,\ M_j^F = j$.

Hence we get 2^{2d} equations linking two groups of states 2^d cardinality each. We call such a construction a *biclique* (a complete bipartite graph in the graph theory) of dimension d. A single biclique of dimension d tests 2^{2d} preimage candidates with only 2^d computations of each chunk. Therefore, a pseudo-preimage attack with a biclique of dimension d has complexity about 2^{n-2d} as long as bicliques can be constructed efficiently. Subsequently we show how the concept of a biclique allows a differential view on the preimage attack and involves techniques from differential cryptanalysis.

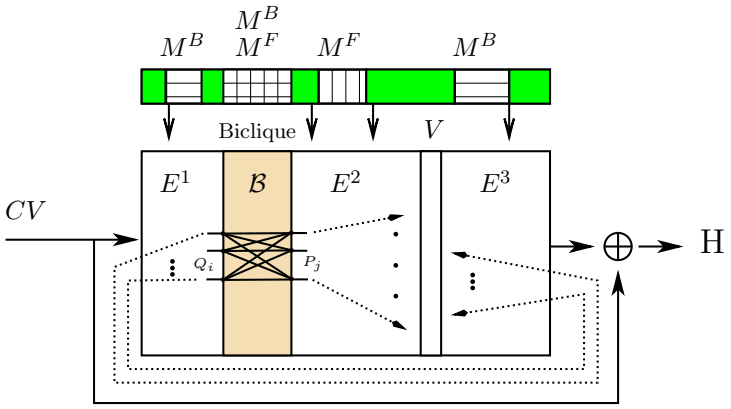

Figure 2: Splice-and-cut attack with a biclique

Differential view. The biclique equations (1) imply that the combinations (Q_i, P_j) are essentially enumerated by pairs (M_i^B, M_j^F). Also, if we start from a P_j with different M^B, we arrive at different Q. Therefore, we get a differential, which starts with zero difference in P and ends with a difference in Q due to the difference ∇M in M^B. The same holds for the difference ΔM in M^F and P:

$$\nabla Q \xleftarrow[\mathcal{B}]{\nabla M||0} 0; \qquad 0 \xrightarrow[\mathcal{B}]{0||\Delta M} \Delta P.$$

Altogether, we get a system of differential trails:

$$\nabla Q \xrightarrow[\mathcal{B}]{\nabla M||\Delta M} \Delta P. \tag{2}$$

Vice versa, a solution to (2) is a solution to (1). As a result, we can find values that yield a biclique as a solution to the system of differentials, and make use of numerous tools from the differential cryptanalysis. Please note that we have to work with several differentials simultaneously, so that even the simplest case $d = 1$ leads to a system of four differentials.

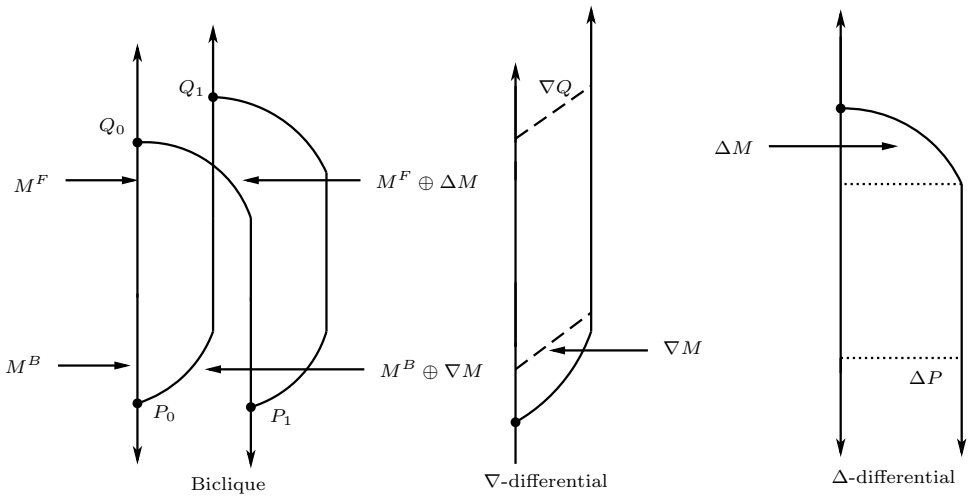

Figure 3: Biclique of dimension 1 and its differentials.

3.2 Biclique construction algorithms

In this section we propose a set of algorithms for constructing bicliques. The choice of the algorithm depends on diffusion properties of the compression function, its message schedule, and applicability of various tools like auxiliary paths, trail backtracking, and message modification.

Algorithm 1. One way to build a biclique is to fix $2^d - 1$ differential trails for the Δ-*differential*:

$$0 \xrightarrow[\mathcal{B}]{0||\Delta M} \Delta P. \tag{3}$$

Then find a solution $\{Q_0; P_0, P_1, \ldots, P_{2^d-1}\}$. Next, construct Q_i as

$$Q_i \xleftarrow[\mathcal{B}]{M_i^B||0} P_0. \tag{4}$$

Assume that each computation does not affect active non-linear elements in the trail (3). Then Q_i conform to this trail, and we get

$$Q_i \xrightarrow[\mathcal{B}]{M_i^B||M_j^F} P_j.$$

To estimate the complexity, assume that the trails (3) altogether have q independent bit conditions on internal variables. Assume also that the computation (4) affects none of these q conditions with probability 2^{-t}. Then the probability that 2^d such computations affect no conditions each is 2^{-t2^d}. Therefore, the biclique is formed with probability 2^{-t2^d}, so we need 2^{t2^d} solutions of Equation (3) to build it (which is feasible for small d). This approach is used in the preimage attack on the hash function Skein-512. Note that the condition of non-interference is equivalent to the fact that trails (3) and

$$\nabla Q \xrightarrow[\mathcal{B}]{\nabla M||0} 0,$$

A similar approach also works in the other direction: consider $2^d - 1$ differential trails for the ∇-*differential*:

$$\nabla Q \xrightarrow[\mathcal{B}]{\nabla M||0} 0,$$

and find a solution $\{P_0; Q_0, Q_1, \ldots, Q_{2^d-1}\}$.

Algorithm 2. (Modification of Algorithm 1 for the case when we can control internal state and message injections within the biclique).

Consider a ∇-differential:

$$\nabla Q \xrightarrow[\mathcal{B}]{\nabla M||0} 0,$$

and choose some $\{Q_0, Q_1, \ldots, Q_{2^d-1}\}$ in accordance with ∇Q. Then we apply the differential

$$0 \xrightarrow[\mathcal{B}]{0||\Delta M} \Delta P.$$

to Q_0 and produce $\{P_0, P_1, \ldots, P_{2^d-1}\}$. The idea is to assign the values to M^C adaptively so that the differentials do not interfere, then the sum of ∇- and Δ-differentials provide us with Equation (2), i.e. with the biclique. In case of a the nonlinear message schedule, an important issue is that the details of ∇- and Δ-trails depend on the value of M^C. The procedure that properly defines those details by pre-fixing some message bits is called message compensation and is in widespread use in the attacks on SHA-2. We use all these techniques in the attacks on SHA-256 and SHA-512.

Algorithm 3. This algorithm is applicable when message injections are large but separated by $r > 1$ rounds, like in Skein. We build a biclique in $2r$ rounds as follows. First, fix the neutral bits and consider $2^d - 1$ differential trails for r-round ∇- and Δ-differentials:

$$\nabla Q \xrightarrow[E_1]{\nabla M||0} 0, \quad 0 \xrightarrow[E_1]{0||\Delta M} \Delta P.$$

Then we aim to find a $2r$-round biclique with a Δ-trail in the first r rounds and a ∇-trail in the last r rounds, so that the trails do not affect each other. We start from the internal state in the message injection and use the freedom in the state and in the message to fulfill conditions in both directions with tools like message modification and auxiliary paths.

4 Simple case: second preimage attack on Skein-512 hash

Skein [10] is the SHA-3 finalist and one of the fastest candidate, which draws a lot of attention of cryptanalysts. Differential [4] and rotational cryptanalysis [15] led the designers to tweak the design twice. As a result, a property that allowed cryptanalyst to penetrate the highest number of rounds, a rotational property, does not exist anymore in the final-round version of Skein, hence the best known attack are near-collisions on up to 24 rounds (rounds 20-43) of the compression function of Skein [4, 26].

The cryptanalysis of the Skein hash function, however, is very limited. Rotational attacks did not extend to the hash function setting, and the differential attacks were not applied in this model. In fact there is no cryptanalytic attack known on any round-reduced version of Skein at all. We subsequently give the first attack in this arguably much more relevant setting.

Since this is the first application of our method, we prefer to give the simplest example in the strongest model rather than attack the highest number of rounds. We consider the Skein-512 hash function reduced to rounds 3-24 (22-round version).

4.1 Description of Skein-512

Skein-512 is based on the block cipher Threefish-512 — 512-bit block cipher with 512-bit key parametrized by 64-bit tweak. Both the internal state I and the key K consist of eight 64-bit words, and the tweak T is two 64-bit words. The compression function $F(CV, T, M)$ of Skein is defined as:

$$F(CV, T, M) = E_{CV,T}(M) \oplus M,$$

where $E_{K,T}(P)$ is the Threefish cipher, CV is the previous chaining value, T is the tweak, and M is the message block. The tweak value is a function of several parameters including the index of the last bit of the message.

Threefish-512 transforms the plaintext P in 72 rounds as follows:

$$P \rightarrow \text{Add subkey } K^0 \rightarrow 4 \text{ rounds} \rightarrow \text{Add } K^1 \rightarrow 4 \text{ rounds} \rightarrow \ldots \rightarrow 4 \text{ rounds} \rightarrow \text{Add } K^{18} \rightarrow C.$$

The subkey $K^s = (K_0^s, K_1^s, \ldots, K_7^s)$ is produced out of the key $K = (K[0], K[1], \ldots, K[7])$ as follows:

$$K_j^s = K[(s+j) \bmod 9], \quad 0 \le j \le 4; \quad K_5^s = K[(s+5) \bmod 9] + T[s \bmod 3];$$
$$K_6^s = K[(s+6) \bmod 9] + T[(s+1) \bmod 3]; \quad K_7^s = K[(s+7) \bmod 9] + s,$$

where s is a round counter, $T[0]$ and $T[1]$ are tweak words, $T[2] = T[0] + T[1]$, and $K[8] = C_{240} \oplus \bigoplus_{j=0}^{7} K[j]$ with a constant C_{240}.

One round transforms the internal state as follows. The eight words $I^0, I^1, \ldots, I^7$ are grouped into pairs and each pair is processed by a simple 128-bit function MIX. Then all the words are permuted by the operation PERM. The details of these operation are irrelevant for the high-level description, for completeness they can be found in Appendix A. We use the following notation for the internal states in round r:

$$S^{r-A} \xrightarrow{MIX} S^{r-M} \xrightarrow{PERM} S^{r-P}$$

4.2 Second preimage attack on Skein-512

In the hash function setting we are given the message M and the tweak value T, and have to find a second preimage. We produce several pseudo-preimages (CV, M') to a call of the compression function that uses 512 bits of M and then find a valid prefix that maps the original IV to one of the chaining values that we generated. Our attack works as follows:

1. Choose the bits $K[2]_{58,59,60}$ as M^F and $K[4]_{55,56,57}$ as M^B. Additionally require $K[2]_{58,59,60} = K[1]_{58,59,60}$ and $K[4]_{55,56,57} = K[3]_{55,56,57}$.

2. Build a biclique of dimension 3 in rounds 12-15 with key additions (key addition + 4 rounds + key addition).

3. Compute forward chunk in rounds 16-19, backward chunks in rounds 8-11, and bits $I^1_{30,31,53}$ of the the state S^{24-P} in both directions in the partial matching procedure.

Biclique					
Rounds	Dimension	M^F bits	M^B bits	Complexity	Freedom used
12-15	3	$K^2_{58,59,60}$	$K^4_{61,62,63}$	2^{200}	200
Chunks		**Matching**			
Forward	Backward	Partial matching	Matching bits	Matching pairs	Complexity
8-11	16-19	$20 \rightarrow 25 = 3 \leftarrow 7$	$I^1_{30,31,53}$	2^3	$2^{2.1}$

Table 2: Parameters of the preimage attack on the 22-round Skein-512

4. Check for the match in these bits, produce 2^3 key candidates, reduce to $2^{2.9}$ due to the type I error. Check them for the match on the full state.

5. Generate a new biclique out of the first one by change of key bits.

6. Repeat steps 2-5 $2^{507.5}$ times and generate $2^{507.5-509+2.9} = 2^{1.6}$ full pseudo-preimages.

7. Match one of the pseudo-preimages with the real IV_0.

Step 2. We construct a biclique with Algorithm 1 (Section 3.2) so that Q_i are internal states S^{11-P} (before the key addition in round 11) and P_j are internal states S^{16-A} (before round 16). The differential trails (3) are built on the propagation of the difference ΔM in M^F in the linearized Skein. We use a version of the 4-round differential trail from the paper [4], which has probability 2^{-68} and, thus, 68 bit conditions if taking the message addition into account. The number of active bits is given in Table 6. For the trail based on the 3-bit difference ΔM we have 197 sufficient conditions since some of them are common. A computation of Q_i out of P_0 do not affect the sufficient conditions with probability $2^{-0.3}$ (checked on a PC). Therefore, for the eight states P_j the probability is $2^{-0.3 \cdot 8} \approx 2^{-3}$. We construct a 4-round biclique with complexity at max $2^{197+3} = 2^{200}$. Note that we have $1024 - 200 = 824$ degrees of freedom left.

Steps 3-4. We have checked experimentally that bits $I^5_{0,1}$ and I^7_{43} of S^{24-P} can be computed from both chunks independently with probability 0.91, so with probability $2^{-0.1}$ we have the type I error and the candidate is discarded.

Step 5. Now we argue that the biclique is essentially independent of the most of M^C bits. Indeed, let us fix the internal states in one of middle rounds, e.g., in round 13, and change M^C. This will affect the key addition before round 12 and the key addition after round 15, and rounds before 12 and after 15. However, these rounds are the part of the chunks, so the equations of the biclique are violated only if the key addition changes its differential behavior. This happens only if carries appear in bit 55 of $K[5], K[4]$ or bit 58 of $K[2], K[1]$. The probability of this event is negligible for most bit flips in M^C. We note that a flip of a single bit in the key makes us to recalculate only a half of rounds 8-11 and 16-19. Furthermore, for the partial matching we need to compute only half of rounds 3-5 and 21-24. Therefore, we have to compute $(4 + 4 + 4 + 4)/2 + 2 + 1 = 11$ full rounds on average. As a result, the amortized cost per bilcique is 2^{-1} calls of the compression function.

Complexity. Each biclique generates eight states for each chunk. In total, we produce 2^3 candidates that match in 3 bits after $8 \cdot 2^{-1} = 2^2$ calls of the Skein-512 compression function. Each candidate is checked for the full match with cost of one round, so the complexity of steps 3-5 is $2^{2.1}$. Taking the type I error into account, we produce a full match with probability $2^{-506.1}$ and complexity $2^{2.1}$. We need $3 = 2^{1.6}$ pseudo-preimages to match one of $2^{510.4}$ prefixes. Therefore, the total complexity is $2^{509.8} + 2^{510.4} = 2^{511.1}$. We also express the main details of the attack in Table 2.

5 Preimage attacks on the SHA-2 family

The SHA-2 family is the object of very intensive cryptanalysis in the world of hash functions. In contrast to its predecessors, collision attacks are no longer the major threat with the best attack on 24 rounds of the hash function [12, 21]. So far the best attacks on the SHA-2 family are preimage attacks on the hash function in the splice-and-cut framework [1] and a boomerang distinguisher that is only applicable for the compression function [16]. We demonstrate that our concept of biclique adds two rounds to the attack on SHA-256, four rounds to the attack on SHA-512, and many more when attacking the compression functions. The number of rounds we obtain for the compression function setting is in both cases comparable to [16], the later however does not allow extension to the hash function nor does it violate any "traditional" security requirement.

5.1 Specification of SHA-2 Family of Hash Functions

We briefly review parts of the specification [20] needed for the cryptanalysis. The SHA-2 hash functions are based on a compression function that updates the state of eight 32-bit state variables $A, \ldots, H$ according to the values of 16 32-bit words $M_0, \ldots, M_{15}$ of the message. SHA-384 and SHA-512 operate on 64-bit words. For SHA-224 and SHA-256, the compression function consists of 64 rounds, and for SHA-384 and SHA-512 — of 80 rounds. The full state in round r is denoted by S^r.

The i-th step uses the i-th word W^i of the expanded message. The message expansion works as follows. An input message is split into 512-bit or 1024-bit message blocks (after padding). The message expansion takes as input a vector M with 16 words and outputs a vector W with n words. The words W^i of the expanded vector are generated from the initial message M according to the following equations (n is the number of steps of the compression function):

$$W^i = \begin{cases} M^i & \text{for } 0 \leq i < 15 \\ \sigma_1(W^{i-2}) + W^{i-7} + \sigma_0(W^{i-15}) + W^{i-16} & \text{for } 15 \leq i < n \end{cases}. \tag{5}$$

where $\sigma_0(x)$ and $\sigma_1(x)$ are linear functions. Other details are irrelevant for the high-level view and are given in the Appendix B.

5.2 Preimage attack on SHA-256

The message schedule of the SHA-2 family is nonlinear, so the number of attacked rounds depends significantly on the position of the biclique. We apply the following reasoning:

- The message injections in rounds 14-15 are partially determined by the padding rules;
- Freedom in the message reduces the biclique amortized cost;
- Chunks do not bypass the feedforward operation due to high nonlinearity of the message schedule;
- There exists a 6-round trail with few conditions easy to use as a ∇-differential.
- Chunks do not have maximal length, otherwise the biclique trail becomes too dense.

Taking these issues into account, we base our attack on a 6-round biclique in rounds 17-22. The full layout is provided in Table 3. The biclique is constructed with Algorithm 2, Section 3.2:

1. Fix a group of 6-round differential trails (the one based on 3-bit difference is listed in Table 7)

$$\nabla Q \xrightarrow[E_1]{\nabla M||0} 0.$$

   Derive the set of sufficient conditions on the internal states (Table 9).

Biclique					
Rounds	Dimension	ΔM bits	∇M bits	Complexity	Freedom used
17-22	3	$W^{17}_{25,26,27}$	$W^{22}_{22,23,31}$	2^{32}	416

Message compensation	
Equations	Constants used in the biclique
9	2

Chunks		Matching		
Forward	Backward	Partial matching	Matching bits	Complexity per match
2-16	23-37	$37 \rightarrow 38 \leftarrow 1$	$A^{38}_{0,1,2,3}$	2^3

Table 3: Parameters of the preimage attack on the 45-round SHA-256

2. Fix the message compensation equations with constants $c_1, c_2, \ldots, c_9$ (Section C).

3. Fix an arbitrary Q_0 and modify it so that most of conditions in the computation $Q_0 \rightarrow P_0$ are fulfilled. Derive Q_i out of Q_0 by applying ∇Q.

4. Fix a group of 2-round trails (the one based on 3-bit difference is given in Table 8) ($\Delta W^{17} \rightarrow \Delta S^{19}$) as a Δ-differential (Equation (3)) in rounds 17-19.

5. Choose $W^{17}, W^{18}, \ldots, W^{22}$ and constants c_8, c_9 so that the conditions in the computations $Q_0 \rightarrow P_j, j = 0, \ldots, 7$ are fulfilled. Produce all P_j.

An algorithm for the biclique is detailed in Appendix, Section C. Finally, we produce $Q_0, \ldots, Q_7$ and $P_0, \ldots, P_7$ that conform to the system (1) and hence form a biclique.

The complexity of building a single biclique is estimated as 2^{32}. However, as many as 7 message words are left undefined in the message compensation equations, which gives us enough freedom to reuse a single biclique up to 2^{256} times. The complexity to recalculate the chunks is upper bounded by 2^2 calls of the compression function. The total amortized complexity of running a single biclique and produced 2^2 matches on 4 bits is 2^3 calls of the compression function (see details in Appendix). Since we need 2^{252} matches, the complexity of the pseudo-preimage search is 2^{253}. Therefore, a full preimage can be found with complexity approximately $2^{1+(253+256)/2} \approx 2^{255.5}$ by restarting the attack procedure $2^{\frac{256-253}{2}} = 2^{1.5}$ times. Memory requirements are approximately $2^{1.5} \times 24$ words.

5.3 Preimage attack on SHA-512

Our attack on SHA-512 does not fix all the 129 padding bits of the last block. This approach still allows to generated short 2nd-preimages by using the first preimage to invest the last block that includes the padding and perform the preimage attack in the last chaining input as the target.

For a preimage attack without a first preimage, expandable messages as e.g. described in [14] can be used. This adds no noticeable cost as the effort for this is only slightly above the birthday bound. In addition, the compression function attack needs to fulfill the following two properties:

Firstly, the end of the message (before the length encoding, i.e., the LSB of W^{13}) has to be '1'. Secondly, the length needs to be an exact multiple of the block length, i.e., fix the last nine bits of W^{15} to "1101111111" (895). In total eleven bits would need to be fixed for this. In the further text we show how to fulfill these conditions.

5.3.1 Attack layout

The basic parameters of the pseudo-preimage attack are given in Table 4. The biclique is constructed by an algorithm similar to the attack on SHA-256 (Algorithm 2, Section 3.2):

Biclique					
Rounds	Dimension	M^F bits	M^B bits	Complexity	Freedom used
21-26	3	$W^{21}_{60,61,62}$	$W^{26}_{53,54,55}$	2^{32}	96

Equations	Message compensation		Constants used in the biclique
9			2

Chunks		Matching		
Forward	Backward	Partial matching	Matching bits	Complexity per match
6-20	27-40	$41 \rightarrow 43 \leftarrow 5$	$A^{43}_{0,1,2}$	2^3

Table 4: Parameters of the preimage attack on the 50-round SHA-512

1. Fix a group of 6-round differential trails (Table 10) for the differential

$$\nabla Q \xrightarrow[\mathcal{B}]{\nabla M||0} 0.$$

   Derive the set of sufficient conditions on the internal states.

2. Fix the message compensation equations with 9 constants (Appendix D).

3. Fix an arbitrary Q_0 and modify it so that the most of conditions in the computation $Q_0 \rightarrow P_0$ are fulfilled. Derive Q_i out of Q_0 by applying ∇Q.

4. Fix a group of 3-round trails (Table 11) ($\Delta W^{21} \rightarrow \Delta S^{23}$) as Δ-differentials (Equation (3)) in rounds 21-23.

5. Choose $W^{21}, W^{22}, \ldots, W^{26}$ and constants c_8, c_9 so that the conditions in the computations $Q_0 \rightarrow P_j, j = 0, \ldots, 7$ are fulfilled. Produce all P_j.

Trail details for the biclique are detailed in Appendix D. Finally, we produce $Q_0, \ldots, Q_7$ and $P_0, \ldots, P_7$ that conform to the system (1) and hence form a biclique.

The complexity of building a single biclique is estimated to be 2^{32} units. However, the amortized cost is again negligible, since we have much freedom in unused message words. The complexity of getting 2^3 matches on 3 bits is 2^3 calls of the compression function (more details in Appendix). Since we need 2^{509} matches, the complexity of the pseudo-preimage search is 2^{509}. Therefore, a full preimage can be found with complexity approximately $2^{1+(509+512)/2} \approx 2^{511.5}$ by restarting the attack procedure $2^{\frac{512-509}{2}} = 2^{1.5}$ times. Memory requirements are approximately $2^{1.5} \times 24$ words.

6 Attacks on the compression functions: SHA-2 and Skein

6.1 Preimage attacks on the Skein compression functions

In this section we provide an attack on the 37-round Skein-512 compression function. In the compression function setting we control the tweak value, which gives us additional freedom both in chunks and the construction of the biclique.

Local collision in Skein-512. If an attacker controls both the IV and the tweak he is able to introduce difference in these inputs so that one of subkeys has zero difference. As a result, he gets a differential which has no difference in internal state for 8 rounds. The lowest weight of input and output differences is achieved in the following combination:

$$\Delta K[6] = \Delta K[7] = \Delta T[1] = \delta,$$

which gives difference $(0, 0, \ldots, 0, \delta)$ in the subkey K^0 and $(\delta, 0, 0, \ldots, 0)$ in K^8, and zero difference in the subkey K^4. The local collisions for further rounds are constructed analogously. We use the

following differences in the compression function attack to make a local collision in rounds 8-15 and 24-31:

$$\Delta K[0] = \Delta T[0] = \Delta T[1] = 1 \ll 63; \qquad \Delta K[3] = \Delta K[4] = \Delta T[1] = 1 \ll 63.$$

Attack. The attack parameters are listed in Table 5. We build a biclique in rounds 24-31, and apply the attack to rounds 2-38, i.e., to the 37-round compression function.

Biclique					
Rounds	Dimension	M^F bits	M^B bits	Complexity	Freedom used
16-23	1	$K[0]$	$K[4]_{63}$	2^{256}	162
Chunks		**Matching**			
Forward	Backward	Partial matching	Matching bit	Matching pairs	Complexity
8-15	24-31	$32 \rightarrow 39 = 2 \leftarrow 7$	I^3_{25}	2^2	$2^{1.1}$

Table 5: Parameters of the preimage attack on the Skein-512 compression function

We use two differential trails: Δ-trail for rounds 16-19 (including key addition in round 19) and ∇-trail for rounds 20-23. The differential trails are based on the evolution of a single difference in the linearized Skein. The Δ-trail has probability 2^{-52}. The ∇-trail has probability 2^{-29}.

The biclique is constructed as follows. First, we restrict to rounds 19-20, where the compression function can be split into two independent 256-bit transformations. A simple approach with table lookups gives a solution to restricted trails with amortized cost 1 (more efficient methods certainly exist). Then we extend this solution to an 8-round biclique by the bits of K^5. We use K^5 in the messagemodification-like process and adjust the sufficient conditions in rounds 16-23. We have 221 degrees of freedom for that (computed on a PC). As many as 96 bits of freedom do not affect the biclique at all and are used to reduce the amortized cost to only a single round.

In the matching part we recompute 29 rounds per biclique. However, a single key bit flip affects only half of rounds 12-15 and 24-27, and also we need to compute only a half of rounds 2-5 and 35-38. In total, we recompute 42 rounds, or $2^{1.2}$ calls of the compression function per structure, and get 2 candidates matching on one bit. The full preimage is found with complexity $2^{511.2}$.

6.2 Preimage attacks on the SHA-2 compression functions

In this section we provide short description of attacks on the SHA-2 compression functions. As long as we do not attack the full hash function, the preimage attack on the compression function is relevant if it is faster than 2^n, though not all these attacks are convertible to the hash function attacks. As a result, we can apply the splice-and-cut attack with the minimum gain to squeeze out the maximum number of rounds. This implies that we consider bicliques of dimension 1. In differential terms, we consider single bit differences ΔM and ∇M. As a result, we get sparse trails with few conditions, and may extend them for more rounds.

- Build 11-round biclique out of a 11-round ∇-trail in rounds 17-27 (SHA-256) and 21-31 (SHA-512). The trail is a variant of the trail in Table 7 that starts with one-bit difference.
- Construct message words in the biclique as follows. In SHA-256 fix all the message words to constants, then apply the difference ΔM to W^{17}, and assume the linear evolution of ΔM when calculating ΔW^{17+i} from $W^2, \ldots, W^{17}$. Assume also the linear evolution of ∇M when calculating ∇W^{27-i} from $W^{28}, \ldots, W^{42}$. Analogously for SHA-512.
- Build the biclique using internal message words as freedom, then spend the remaining 5 message words to ensure the Δ and ∇-trails in the message schedule. As a result, we get the longest possible chunks (2-16 and 28-42 in SHA-256).

Therefore, we gain 5 more rounds in the biclique, and two more rounds in the forward chunk. This results in a 52-round attack on the SHA-256 compression function, and a 57-round attack on the SHA-512 compression function.

7 Discussion and Conclusions

We reconsidered meet-in-the-middle attacks as a means to obtain cryptanalytically significant results on narrow-pipe hash functions like SHA-256 and SHA-512, and the SHA-3 finalist Skein. We have demonstrated that the concept of the biclique is the source for improvements on preimage attacks, which brings up to ten rounds more in the attack depending on the model. To summarize, we benefit from the following features:

- Replacement of the initial structure idea with a formal concept of biclique and its convergence to differential cryptanalysis;
- Use of differential trails in a biclique with a small number of sufficient conditions;
- Deterministic algorithms to build a biclique, which can be adapted for a particular primitive;
- Use of various tools from differential cryptanalysis like trail backtracking [5], message modification and neutral bits [6, 13, 19, 27], and rebound techniques [18].

Overall, the differential view gives us much more freedom and flexibility compared to previous attacks. Though all the functions in this paper are ARX-based, our technique can be as well applied to other narrow-pipe designs.

For SHA-256, SHA-512, and Skein-512, we considered both the hash function and the compression function setting. In all settings we obtained cryptanalytic results on more rounds than any other known method. Using these data points, it seems safe to conclude that Skein-512 is more resistant against splice-and-cut cryptanalysis than SHA-512. An interesting problem to study would be possibilities for meaningful bounds on the length of biclique structures.

Acknolwedgements

Part of this work was done while Christian Rechberger was with KU Leuven and visiting MSR Redmond. This work was supported by the European Commission under contract ICT-2007-216646 (ECRYPT II).

References

[1] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In *ASIACRYPT'09*, volume 5912 of *LNCS*, pages 578–597. Springer, 2009.

[2] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *Selected Areas in Cryptography'08*, volume 5381 of *LNCS*, pages 103–119. Springer, 2008.

[3] Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *CRYPTO'09*, volume 5677 of *LNCS*, pages 70–89. Springer, 2009.

[4] Jean-Philippe Aumasson, Çagdas Çalik, Willi Meier, Onur Özen, Raphael C.-W. Phan, and Kerem Varici. Improved cryptanalysis of Skein. In *ASIACRYPT'09*, volume 5912 of *LNCS*, pages 542–559. Springer, 2009.

[5] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. RadioGatun, a belt-and-mill hash function. *NIST Cryptographic Hash Workshop, available at* `http://radiogatun.noekeon.org/`, 2006.

[6] Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[7] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In *CRYPTO'92*, volume 740 of *LNCS*, pages 487–496. Springer, 1992.

[8] Christophe De Cannière and Christian Rechberger. Preimages for reduced SHA-0 and SHA-1. In *CRYPTO'08*, volume 5157 of *LNCS*, pages 179–202. Springer, 2008.

[9] David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *LNCS*, pages 192–211. Springer, 1985.

[10] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein hash function family (version 1.3, 1 Oct, 2010).

[11] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In *ASIACRYPT'10*, volume 6477 of *LNCS*, pages 56–75. Springer, 2010.

[12] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and Other Non-random Properties for Step-Reduced SHA-256. In *Selected Areas in Cryptography'08*, volume 5381 of *LNCS*, pages 276–293. Springer, 2008.

[13] Antoine Joux and Thomas Peyrin. Hash functions and the (amplified) boomerang attack. In *CRYPTO'07*, volume 4622 of *LNCS*, pages 244–263. Springer, 2007.

[14] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.

[15] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational Rebound Attacks on Reduced Skein. In *ASIACRYPT'10*, volume 6477 of *LNCS*, pages 1–19. Springer, 2010.

[16] Mario Lamberger and Florian Mendel. Higher-order differential attack on reduced SHA-256. available at `http://eprint.iacr.org/2011/037.pdf`, 2011.

[17] Gaëtan Leurent. MD4 is not one-way. In *FSE'08*, volume 5086 of *LNCS*, pages 412–428. Springer, 2008.

[18] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE'09*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.

[19] Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, and Kazuo Ohta. Improved collision search for SHA-0. In *ASIACRYPT'06*, volume 4284 of *LNCS*, pages 21–36. Springer, 2006.

[20] NIST. FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.

[21] Somitra Kumar Sanadhya and Palash Sarkar. New collision attacks against up to 24-step SHA-2. In *INDOCRYPT'08*, volume 5365 of *LNCS*, pages 91–103. Springer, 2008.

	I^0	I^1	I^2	I^3	I^4	I^5	I^6	I^7	Conditions in the round
S^{12-A}								3	3
S^{13-A}				6	3				9
S^{14-A}	6		3			3		12	24
S^{15-A}	3	6	3	24	12	6	6	3	161

Table 6: Number of active bits in the most dense Δ-trail in 4 rounds of Skein-512.

[22] Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In *ASIACRYPT'08*, volume 5350 of *LNCS*, pages 253–271. Springer, 2008.

[23] Yu Sasaki and Kazumaro Aoki. Preimage attacks on step-reduced MD5. In *ACISP'08*, volume 5107 of *LNCS*, pages 282–296. Springer, 2008.

[24] Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In *EUROCRYPT'09*, volume 5479 of *LNCS*, pages 134–152. Springer, 2009.

[25] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In *EUROCRYPT'07*, volume 4515 of *LNCS*, pages 1–22. Springer, 2007.

[26] Bozhan Su, Wenling Wu, Shuang Wu, and Le Dong. Near-Collisions on the Reduced-Round Compression Functions of Skein and BLAKE. Cryptology ePrint Archive, Report 2010/355, 2010. `http://eprint.iacr.org/`.

[27] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *CRYPTO'05*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[28] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[29] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The second-preimage attack on MD4. In *CANS'05*, volume 3810 of *LNCS*, pages 1–12. Springer, 2005.

A More details on Skein specification

The operation MIX has two inputs x_0, x_1 and produces two outputs y_0, y_1 with the following transformation:

$$y_0 = x_0 + x_1$$
$$y_1 = (x_1 \lll_{R_{(d \bmod 8)+1,j}}) \oplus y_0$$

The exact values of the rotation constants $R_{i,j}$ as well the permutations π (which are different for each version of Threefish) can be found in [10].

B More details on SHA-2 specification

The round function of all the SHA-2 functions operates as follows:

$$T_1^{(i)} = H^i + \Sigma_1(E^i) + \mathrm{Ch}(E^i, F^i, G^i) + K^i + W^i,$$
$$T_2^{(i)} = \Sigma_0(A_i) + \mathrm{Maj}(A_i, B_i, C_i),$$
$$A^{i+1} = T_1^{(i)} + T_2^{(i)}, B^{i+1} = A^i, C^{i+1} = B^i, D^{i+1} = C^i,$$
$$E^{i+1} = D^i + T_{(i)}^1,\ F^{i+1} = E^i,\ G^{i+1} = F^i,\ H^{i+1} = G^i.$$

Here K^i is a round constant.The round function uses the bitwise boolean functions Maj and Ch, and two GF(2)-linear functions $\Sigma_0(x)$ and $\Sigma_1(x)$. Functions Maj and Ch are defined identically for all the SHA-2 functions:

$$\mathrm{Ch}(x,y,z) = x \wedge y \oplus \overline{x} \wedge z \quad (6)$$

$$\mathrm{Maj}(x,y,z) = x \wedge y \oplus x \wedge z \oplus y \wedge z \quad (7)$$

For SHA-224 and SHA-256, $\Sigma_0(x)$ and $\Sigma_1(x)$ are defined as follows:

$$\Sigma_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22), \qquad \Sigma_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25).$$

For SHA-384 and SHA-512, they are defined as follows:

$$\Sigma_0(x) = (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39), \qquad \Sigma_1(x) = (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41).$$

Operations $\ggg$ and $\gg$ denote bit-rotation and bit-shift of A by x positions to the right respectively. The message schedule functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows for SHA-224 and SHA-256:

$$\sigma_0(x) = \quad (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3), \qquad \sigma_1(x) = \quad (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)\,.$$

and for SHA-384 and SHA-512:

$$\sigma_0(x) = \quad (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7)\,, \qquad \sigma_1(x) = \quad (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6)\,.$$

C Details on the 46-round SHA-256 attack

Message compensation. Since any consecutive 16 message words in SHA-2 bijectively determine the rest of the message block used at an iteration of compression function, we need to place the initial structure within a 16-round block and define such restrictions on message dependencies that maximize the length of chunks.

We use a heuristic algorithm to check how many steps forward and backward can be calculated independently with a 6-step initial structure. We discovered that with W^{17} and W^{22} selected as the words with neutral bits, it is possible to expand 16-round message block $\{W^{12},\ldots,W^{27}\}$ by 10 steps backwards and 9 steps forwards, so that $\{W^2,\ldots,W^{16}\}$ are calculated independently of W^{17} , and $\{W^{23},...,W^{36}\}$ are calculated independently of W^{22}. Below we define the message compensation conditions that make such chunk separation possible (neutral bit words are outlined in frames):

$$\begin{array}{lll} -\sigma_1(W^{25}) + W^{27} = c_1; & -W^{19} - \sigma_1(W^{24}) + W^{26} = c_2 & -\sigma_1(W^{23}) + W^{25} = c_3 \\ -\boxed{W^{17}} + W^{24} = c_4 & -\sigma_1(W^{21}) + W^{23} = c_5; & -\sigma_1(W^{19}) + W^{21} = c_6 \\ -\sigma_1(\boxed{W^{17}}) + W^{19} = c_7; & W^{12} + \sigma_0(W^{13}) = c_8; & W^{13} + \boxed{W^{22}} = c_9 \end{array} \quad (8)$$

Fig. 4 explains how the message compensation dependencies are constructed. Columns and rows correspond to message words and equations respectively, where X at the intersection of row i ad column j shows that W^j is a part of i^{th} equation. Colour of a column reflects whether the appropriate message word is set independently of both words with neutral bits (white), calculated using NW^1 (blue) or NW^2 (yellow). We start with $\{W^2,\ldots,W^{11},W^{22}\}$ colored blue and $\{W^{17},W^{28},\ldots,W^{36}\}$ colored yellow (Fig. 4, a) and aim to get rid of equations that involve both 'blue' and 'white' message words. We split these equations and introduce constants $\{c_1,\ldots,c_8,c_9\}$ (in other words, we create additional dependencies between controlled messages and words with neutral bits as shown in Fig. 4, b).

It is easy to see that words $W^{14},\ldots,W^{16},W^{18}$, and W^{20} can be chosen independently of both W^{17} and W^{22}, so we can assign W^{14} and W^{15} with 64-bit length of the message to satisfy padding rules (additionally, 1 bit of W^{13} needs to be fixed). W^{18} and W^{20} are additional freedom for constructing the biclique.

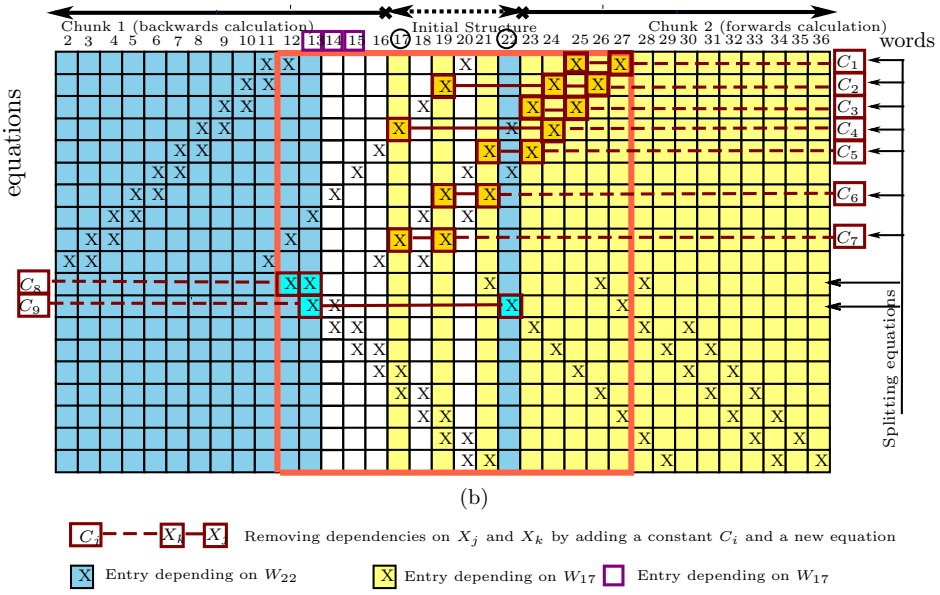

Figure 4: Message dependencies: (a) before and (b) after message compensation in SHA-256

Construction of the biclique. The basic differential trail for the biclique is a 6-round trail in the backward direction ($\Delta_Q \leftarrow \nabla M$) that starts with the difference in bits 22, 23, and/or 31 in W_{22}. The trail is briefly depicted in Table 7 with references to the sufficient conditions (which work out for all the 7 possible differences) in Table 9.

Round	A	B	C	D	E	F	G	H	W	Cond-s
17	-	-	22,23,31	-	-	Λ'	-	*	-	1
18	-	-	-	22,23,31	-	-	Λ'	-	-	3,4
19	-	-	-	-	22,23,31	-	-	Λ'	-	7-11
20	-	-	-	-	-	22,23,31	-	-	-	12
21	-	-	-	-	-	-	22,23,31	-	-	13
22	-	-	-	-	-	-	-	22,23,31	-	
23	-	-	-	-	-	-	-	-	22,23,31	

Table 7: Details for biclique in SHA-256. Differential ∇-trail (active bits). $\Lambda' = \{6, 11, 12, 16, 17, 20, 23, 24, 29, 30\}$

Round	A	B	C	D	E	F	G	H	Cond-s
18	*	-	-	-	25,26,27	-	-	-	2
19	*	*	-	-	Φ	25,26,27	-	-	5,6

Table 8: Details for biclique in SHA-256. Differential Δ-trail (active bits).
$\Phi = \Sigma_1\{25, 26, 27\} = \{0, 1, 2, 14, 15, 16, 19, 20, 21\}$, $*$ stands for arbitrary difference.

We also use bits 25, 26, 27 as neutral in W_{17}. To prevent this difference to interleave with the backward trail difference in round 19, we restrict the behavior of the forward trail as specified in Table 8. The aggregated conditions, which make each forward trail keep the backward ones unaffected, are given in Table 9.

With three neutral bits we construct a biclique with 8 starting points for chunks in each direction. First, we choose the initial state $A_{17}, \ldots, H_{17}$ so that the conditions 1 and 5 are fulfilled. Then we proceed with a standard trail backtracking procedure modifying the starting state if needed. Here

we are free to use all the tools from the collision search like message modification or tunnels. Next, in round 18 we further check whether the value of E stops carries in the forward trail. If not, we change the value of D in the starting state accordingly. Then we sequentially modify the initial state in order to fulfill the conditions 2-11.

The last two conditions are affected by the message words W_{19} and W_{20}. We need to fulfill three bit conditions for every W_{17}, used in the attack. Therefore, we spend $3 \cdot 8 \cdot 2 = 48$ degrees of freedom in message words $W_{17}, W_{18}, W_{19}, W_{20}, W_{21}$. Note that there is a difference in W_{19} determined by the difference in W_{17} due to the message compensation. We have fixed the constants c_6 and c_7 from Eq. 8 while defining W_{19} and W_{21}. In total, we construct the biclique in about 2^{32} time required to find proper W_{19} and W_{20}.

Amount of freedom used. In total, we have 512 degrees of freedom in the message and 256 degrees of freedom in the state. The biclique is determined by the state in round 17 and message words W_{17}–W_{21}. The choice of W_{19} and W_{21} is equivalent to the choice of constants c_6, c_7 in Eq. 8. Therefore, we spend $256 + 5 \cdot 32 = 416$ degrees of freedom for the biclique fulfilling as few as $47 + 42$ (Table 9) conditions. We note that we have more than 300 degrees of freedom left in the construction of a biclique. After the biclique is fixed, there are $768 - 416 = 352$ degrees of freedom left. We spend $32 + 32 + 2 = 66$ for the padding, thus leaving with 286 degrees of freedom. Therefore, one biclique is enough for the full attack.

Round	Conditions	Purpose	F	C	D_W
17	1: $A^{22,23,31} = B^{22,23,31}$	Absorption (MAJ)	IC	3	0
	2 : $(W^{\oplus} E_{18})^{25,26,27} = 0$	Stop forw. carry	SM	6	0
18	3 : $E^{\Lambda'} = 1$,	Absorption (IFF)	SM	9	0
	4 : $(D \oplus E_{19})^{22,23,31} = 0$	Stop carry	SM	3	0
	5 : $F^{25,26,27} = G^{25,26,27}$,	Absorption (IFF)	IC	9	0
	6 : $(S1 \oplus E_{19})^{\Phi} = 0$	Stop forw. carry	SM	2	0
19	7: $F^{22,31} = G^{22,31}$	Absorption (IFF)	SM	2	0
	8: $F^{23} \neq G^{23}$	Pass (IFF)	SM	1	0
	9: $CH^{25} \neq S1^{25}$	Force carry (H)	SM	1	0
	10: $(S1 \oplus H)^{\Lambda} = 1$	Stop carry (H)	SM	9	0
	11: $(CH \oplus H)^{24} = 0$	Force carry (H)	SM	1	0
	11': $(CH \oplus H)^{23} = 0$	Force carry (H)	SM	1	0
20	12 : $E^{22,23,31} = 0$	Absorption (IFF)	W^{19}	21	21
21	13 : $E^{22,23,31} = 1$	Absorption (IFF)	W^{20}	21	21

Table 9: Sufficient conditions for the ∇-trails in SHA-256.
A^i – i-th bit of A. F – how the conditions are fulfilled (IC – initial configuration, SM – state modification).
C – total number of independent conditions. D_W – conditions fulfilled by message words.
$\Lambda = \Sigma_1\{22, 23, 31\} = \{6, 11, 12, 16, 17, 20, 25, 29, 30\}$

D Details on the 50-round SHA-512 attack

Message compensation. The system of compensation equations is defined similarly to the attack on SHA-256:

$$-\sigma_1(W^{29}) + W^{31} = c_1; \qquad -W^{23} - \sigma_1(W^{28}) + W^{30} = c_2; \qquad -\sigma_1(W^{27}) + W^{29} = c_3$$
$$-\boxed{W^{21}} + W^{28} = c_4; \qquad -\sigma_1(W^{25}) + W^{27} = c_5; \qquad -\sigma_1(W^{23}) + W^{25} = c_6$$
$$-\sigma_1(\boxed{W^{21}}) + W^{23} = c_7; \qquad W^{16} + \sigma_0(W^{17}) = c_8; \qquad W^{17} + \boxed{W^{26}} = c_9$$

To satisfy padding rules, we need to use 1 LSB of W^{13} and 10 LSB of W^{15} (see Section 5 for more details about the padding). The choice of constants c_8, c_9 and fixed lower 53 bits of W^{26} provide us with sufficient freedom. Indeed, by choosing c_9 we define lower 53 bits of W^{17}. Having c_8 chosen, we derive 45 lower bits of W^{16} fixed due to σ_0 in message schedule. Further, we get lower 37 bits of W^{15}, 29 bits of W^{14} and 21 bit of W^{13} fixed. As we need only one LSB of W^{13} and 10 LSB of W^{15} to be fixed, we use only lower 33 bits of W^{26}, lower 33 bits of c_9, and lower 25 bits of c_8.

Construction of the biclique for SHA-512. The basic differential trail for the biclique is a 6-round trail in the backward direction ($\Delta_Q \leftarrow \nabla M$) that starts with the difference in bits 53, 54, and/or 55 in W^{26}. The trail is depicted in Table 10 with the number of independent sufficient conditions.We also use bits 60, 61, 62 as neutral in W^{21}. To prevent this difference to interleave with the backward trail difference in round 19, we restrict the behavior of the forward trail as specified in Table 11. Note that the trails based on the linearized version are now compatible with our choice of neutral bits. The biclique is basically the same as in SHA-256, with a small difference that we spend $48 + 48 = 96$ degrees of freedom inside.

Round	A	B	C	D	E	F	G	H	Indep. cond-s
21	-	-	53,54,55	-	-	Λ	-	*	3
22	-	-	-	53,54,55	-	-	Λ	-	12
23	-	-	-	-	53,54,55	-	-	Λ	12
24	-	-	-	-	-	53,54,55	-	-	24
25	-	-	-	-	-	-	53,54,55	-	24
26	-	-	-	-	-	-	-	53,54,55	

Table 10: Biclique in SHA-512. Differential ∇-trail (active bits). $\Lambda = \Sigma_1\{53, 54, 55\} = \{12, 13, 14, 35, 36, 37, 39, 40, 41\}$

Round	A	B	C	D	E	F	G	H	Cond.
22	*	-	-	-	60,61,62	-	-	-	3
23	*	*	-	-	Φ	60,61,62	-	-	18

Table 11: Biclique in SHA-512. Differential Δ-trail.
$\Phi = \Sigma_1\{60, 61, 62\} = \{17, 20, 21, 42, 43, 44, 46, 47, 48\}$, $*$ stands for arbitrary difference.

Complexity estimate. We get a pseudo-preimage with complexity approximately $2^{506} \times 2^3 = 2^{509}$ compression function operations. Therefore, a full preimage can be found with complexity approximately $2^{1+(509+512)/2} \approx 2^{511.5}$ by restarting the attack procedure $2^{\frac{512-509}{2}} = 2^{1.5}$ times from step 2. Memory requirements are approximately 4 message words (2 message words for storing the fixed parts of neutral bits, 2^3 entries of 3 neutral bits difference and 3 bits for matching in each list). For finding a preimage, we need to store $2^{1.5}$ pseudo-preimages, i.e. the memory requirement is $2^{1.5} \times 24$ words.

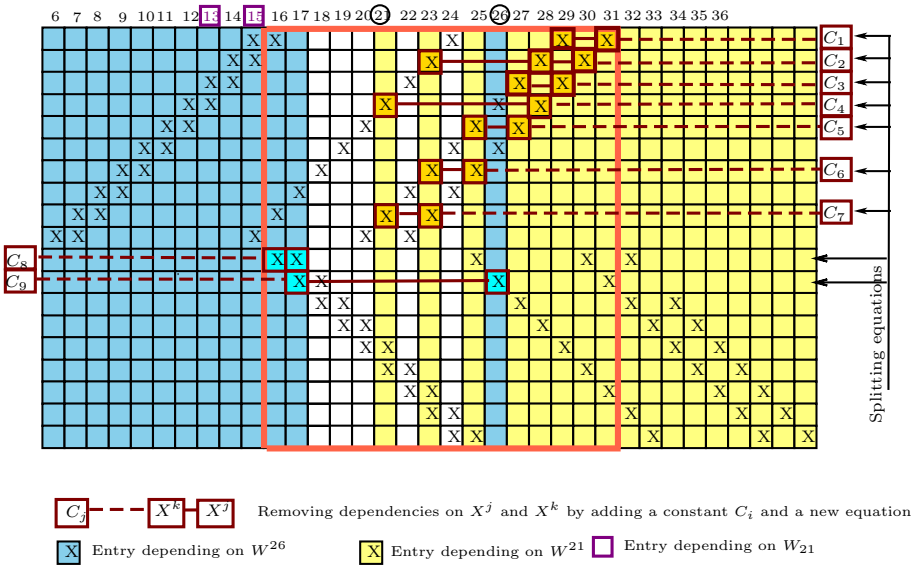

Figure 5: Message compensation in SHA-512