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

Dmitry Khovratovich[1], Christian Rechberger[2], and Alexandra Savelieva[3]

[1] Microsoft Research Redmond, USA
[2] DTU MAT, Denmark
[3] National Research University Higher School of Economics, Russia

Abstract. We present a new concept of biclique as a tool for preimage attacks, which employs many powerful techniques from differential cryptanalysis of block ciphers and hash functions.

The new tool has proved to be widely applicable by inspiring many authors to publish new results of the full versions of AES, KASUMI, IDEA, and Square. In this paper, we show how our concept leads to the first cryptanalysis of the round-reduced Skein hash function, and describe an attack on the SHA-2 hash function with more rounds than before.

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

Major breakthrough in preimage attacks on hash functions happened in 2008 when the so-called *splice-and-cut* framework was introduced. Its applications to MD4 and MD5 [2,25], and later to Tiger [11] brought amazing results. Internal properties of message schedule appeared to be limiting application of this framework to SHA-x family [1,3]. However, the concept of *biclique* introduced in this paper allows to mitigate such obstacles, as demonstrated by our results.

This is the first work on the concept of biclique cryptanalysis. We focus on new definitions and algorithms and concentrate on the hash function setting. As applications, we present an attack on the Skein hash function (the only one existing so far) and the SHA-2 family . Our findings are summarized in Table 1.

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 (called *neutral bits*). If the property holds, the computation of this part remains the same when we flip those bits in the other

* This work was supported by the European Commission under contract ICT-2007-216646 (ECRYPT II) and the Federal Target Program "Scientific and scientific-pedagogical personnel of innovative Russia" in 2009-2013 under contract No. P965 from 27 May, 2010.

A. Canteaut (Ed.): FSE 2012, LNCS 7549, pp. 244–263, 2012.

© International Association for Cryptologic Research 2012

part of a primitive. Assume that neutral 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 and subsequently checks for a match in the middle. Efficiency of the attack depends on the number of neutral bits.

Sasaki and Aoki observed [2,25] that compression functions with permutation-based message schedule are vulnerable to this kind of attack as chunks can be long. They proposed several ways to improve splice-and-cut framework, including a very interesting trick referred to as *initial structure* [3,26]. It can be informally defined as an overlapping of chunks, where neutral bits, although formally belonging to both chunks, are involved in the computation of the proper chunk only. In our work we aimed to explore the potential of initial structure.

Our Contributions. We replace the idea of the initial structure with a more formal and general concept of *biclique* that provides us with a new level of understanding. In terms of graph theory, bicliques are structures represented by two sets of states with each state having a relation with all states in another set. We derive a system of functional equations linking internal states several rounds apart, and show that it is equivalent to a system of differentials, so the full structure of states can be built out of a set of trails. The differential view allows us to apply numerous tools from collision search and differential cryptanalysis. We propose three generic and flexible algorithms for constructing the bicliques.

Our simple example of biclique application is an attack on round-reduced Skein-512 hash function, the SHA-3 finalist which currently lacks any other attacks in the hash setting. We penetrate 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 covers 37 rounds.

Our second group of applications is the SHA-2 family. We heavily use differential trails in SHA-2, message modification techniques from SHA-1 and SHA-0, trail backtracking techniques from RadioGatun, Grindahl, SHA-1, etc., to build attacks on 45-round SHA-256 and 50-round SHA-512 (both the best attacks in the hash mode). For the compression functions, we penetrate up to 7 more rounds, reaching 52 rounds and violating the security of about 80% of SHA-256.

Table 1. New (second) preimage attacks on Skein-512 and the SHA-2 family

Reference	Target	Steps	Complexity			Memory
			Pseudo-preimage	Second Preimage	Preimage	(words)
Section 4	Skein-512	22	2^{508}	2^{511}	-	2^{6}
Section 6	Skein-512	37	$2^{511.2}$	-	-	2^{64}
[1,11]	SHA-256	43	$2^{251.9}$	$2^{254.9}$	$2^{254.9}$	2^{6}
Section 5	SHA-256	45	2^{253}	$2^{255.5}$	$2^{255.5}$	2^{6}
Section 6	SHA-256	52	2^{255}	-	-	2^{6}
[1,11]	SHA-512	46	2^{509}	$2^{511.5}$	$2^{511.5}$	2^{6}
Section 5	SHA-512	50	2^{509}	$2^{511.5}$	$2^{511.5}$	2^{4}
Section 6	SHA-512	57	2^{511}	-	-	2^{6}

2 Bicliques

In this section we introduce preimage attacks with bicliques. We consider hash functions with block cipher based compression functions $H = E_N(X) \oplus X$, where E is the block cipher keyed with parameter N (notation $\xrightarrow{N}$ and $\xleftarrow{N}$ will be used for E computed in forward and backward direction respectively). Depending on the design, parameters (N, X) will be either (M, CV) for the most popular Davies-Meyer mode, or (CV, M) for Matyas-Meyer-Oseas mode, where CV is the chaining variable and M is the message.

Let f be a sub-cipher of E, and $\mathcal{N} = \{N[i,j]\}$ be a group of parameters for f. Then a *biclique of dimension* d over f for $\mathcal{N}$ is a pair of sets $\{Q_i\}$ and $\{P_j\}$ of 2^d states each such that

$$Q_i \xrightarrow[f]{N[i,j]} P_j. \tag{1}$$

A biclique is used in the preimage search as follows (Figure 1). First, we note that if $N[i,j]$ yeilds a preimage, then

$$E: \quad X \xrightarrow{N[i,j]} Q_i \xrightarrow[f]{N[i,j]} P_j \xrightarrow{N[i,j]} H.$$

An adversary selects a variable v outside of f (w.l.o.g. between P_j and H) and checks, for appropriate choices of sub-ciphers g_1 and g_2, if

$$\exists i, j: \ P_j \xrightarrow[g_1]{N[i,j]} v \overset{?}{=} v \xleftarrow[g_2]{N[i,j]} Q_i.$$

A positive answer yields a candidate preimage. Here, to compute v from Q_i, the adversary first computes X and then derives the output of E as $X \oplus H$.

To benefit from the meet-in-the-middle framework, the variable v is chosen so that g_1 and g_2 are independent of i and j respectively:

$$P_j \xrightarrow[g_1]{N[*,j]} v \overset{?}{=} v \xleftarrow[g_2]{N[i,*]} Q_i.$$

Then the complexity of testing 2^{2d} messages for preimages is computed as follows:

$$C = 2^d(C_{g_1} + C_{g_2}) + C_{bicl} + C_{recheck},$$

where C_{bicl} is the biclique construction cost, and $C_{recheck}$ is the complexity of rechecking the remaining candidates on the full state. We explain how to amortize the biclique construction in the next section. Clearly, one needs 2^{n-2d} bicliques of dimension d to test 2^n parameters.

3 Biclique Construction Algorithms

Here we introduce several algorithms for the biclique construction. They differ in complexity and requirements to the dimension of a biclique and properties of the mapping f.

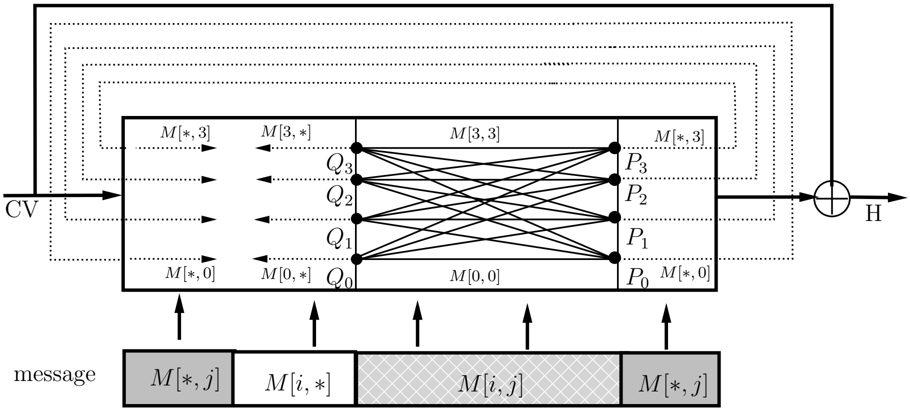

Fig. 1. Biclique of dimension 2 in the meet-in-the-middle attack on a Davies-Meyer compression function

Consider a single mapping in Equation (1)

$$Q_0 \xrightarrow[f]{N[0,0]} P_0. \tag{2}$$

We call this a *basic computation*. Consider the other mappings as differentials to the basic computation:

$$\nabla_i \xrightarrow[f]{\Delta^N_{i,j}} \Delta_j, \tag{3}$$

so that

$$Q_i = Q_0 \oplus \nabla_i, \quad P_j = P_0 \oplus \Delta_j, \quad N[i,j] = N[0,0] \oplus \Delta^N_{i,j}.$$

Vice versa, if a computation (2) is a solution to 2^{2d} differentials in (3), then it is a basic computation for a biclique.

The algorithms presented below allow us to reduce the number of differentials needed for a biclique, and hence construct a biclique efficiently.

Algorithm 1. Let the differences in the set $\mathcal{N}$ be defined as the following linear function:

$$\Delta^N_{i,j} = \Delta^N_j \oplus \nabla^N_i \tag{4}$$

Let us fix Q_0 and construct P_j as follows:

$$Q_0 \xrightarrow[f]{N[0,j]} P_j. \tag{5}$$

As a result, we get a set of trails:

$$0 \xrightarrow[f]{\Delta^N_j} \Delta_j. \tag{6}$$

Let us also construct Q_i out of P_0:

$$Q_i \xleftarrow[f]{N[i,0]} P_0, \tag{7}$$

and get another set of trails:

$$\nabla_i \xleftarrow[f]{\nabla_i^N} 0. \tag{8}$$

Suppose that the trails (8) do not affect active non-linear elements in the trails (6). Then Q_i are solutions to the trails (6), so we get the biclique equation:

$$Q_i \xrightarrow[f]{N[i,j]} P_j. \tag{9}$$

Assume that the computation (7) does not affect active non-linear elements in the trails (6) with probability 2^{-t}. The probability that 2^d computations affect no condition is 2^{-t2^d}. Equation (9) is satisfied with probability 2^{-t2^d}, so we need 2^{t2^d} solutions to Equation (6) to build a biclique (feasible for small d).

This algorithm is used in the preimage attack on the hash function Skein-512. For non-ARX primitives with predictable diffusion it can even be made deterministic. For example, for AES [8] and Square it is easy to build truncated differential trails that do not share active non-linear components with probability 1. As a result, the biclique construction can be simply explained using a picture of trails (Figure 2). ■

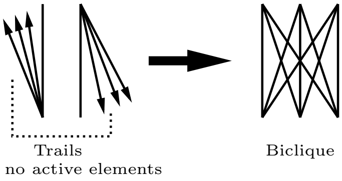

Fig. 2. Biclique construction out of non-interleaving trails

Algorithm 2. (Modification of Algorithm 1 for the case when the hash function operates in DM mode, and we can control internal state and injections of message M within the biclique). Assume that the mapping f uses several independent parts (blocks) of message M via message injections (like in SHA-2). Consider a message group with property (4) but do not define the messages yet. Choose a state Q_0 satisfying sufficient conditions to build sets of trails (6) and (8) that do not share active non-linear components. Then find $N[0,0]$ such that $Q_0 \xrightarrow[f]{N[0,0]} P_0$ conforms to both sets of trails. Since the sets do not share active non-linear components, we get

$$Q_i \xrightarrow[f]{N[i,j]} P_j,$$

where $Q_i = Q_0 \oplus \nabla_i, \quad P_j = P_0 \oplus \Delta_j$.

We can control message injections in f, and therefore, are able to define $N[0,0]$ block by block similarly to the trail backtracking [5]. The procedure that ensures that the message $N[0,0]$ is well-defined, and the trails (6) and (8) do not contradict, was first proposed in [1] and referred to as *message compensation.* ■

Algorithm 3. (for bicliques of dimension 1) We apply this rebound-style [20] algorithm if the mapping f is too long for differential trails with reasonable number of sufficient conditions. Then we split it into two parts f_1 and f_2 and consider two differential trails with probabilities p and q, respectively:

$$0 \xrightarrow[f_1]{\Delta^N} \Delta, \quad \nabla \xrightarrow[f_2]{\nabla^N} 0. \tag{10}$$

We fix the state S between f_1 and f_2, and consider a quartet of states:

$$S,\ S \oplus \Delta,\ S \oplus \nabla,\ S \oplus \Delta \oplus \nabla.$$

Suppose that a quartet of states is a quartet in the middle of the boomerang attack, which happens with probability p^2q^2 for a random N under an approriate independency assumption. Then we derive input states Q_0, Q_1 and output states P_0, P_1, which are linked as follows (see also Figure 3):

$$\begin{aligned} Q_0 \xrightarrow[f_1]{N} S \xrightarrow[f_2]{N} P_0; &\qquad Q_0 \xrightarrow[f_1]{N\oplus\Delta^N} S \oplus \Delta \xrightarrow[f_2]{N\oplus\Delta^N} P_1; \\ Q_1 \xrightarrow[f_1]{N\oplus\nabla^N} S \oplus \nabla \xrightarrow[f_2]{N\oplus\nabla^N} P_0; &\qquad Q_1 \xrightarrow[f_1]{N\oplus\Delta^N\oplus\nabla^N} S \oplus \Delta \oplus \nabla \xrightarrow[f_2]{N\oplus\Delta^N\oplus\nabla^N} P_1. \end{aligned}$$

Therefore, we get a biclique, where the set of parameters $\mathcal{N}$ is defined as follows:

$N[0,0] = N;\ N[0,1] = N \oplus \Delta^N;\ N[1,0] = N \oplus \nabla^N;\ N[1,1] = N \oplus \Delta^N \oplus \nabla^N$.■

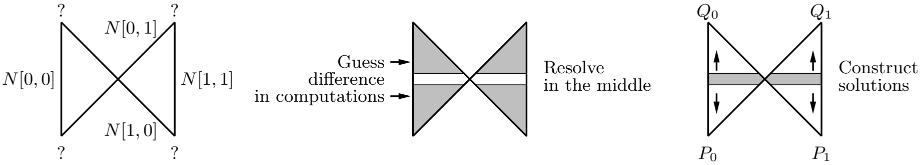

Fig. 3. Rebound-style algorithm for biclique construction

We use Algorithm 2 in the attacks on SHA-256 and SHA-512, and Algorithm 3 is applied in the preimage attack on the Skein compression function. In practice, we use freedom in the internal state and in the message injection fulfill conditions in both trails with tools like message modification and auxiliary paths.

4 Simple Case: Second Preimage Attack on Skein-512

Skein [10] is a SHA-3 finalist, and gets a lot of cryptanalytic attention. Differential [4] and rotational cryptanalysis [17] led the authors of Skein to tweak the design twice. As a result, the rotational property, which allowed cryptanalysts to penetrate the highest number of rounds, does not exist anymore in the new 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 [4,27]. Very recently near-collisions attacks on up to 32 rounds of Skein-256 were demonstrated [29].

The cryptanalysis of Skein in the hash setting is very limited. Rotational attacks did not extend to the Skein hash function, and the differential attacks were not applied in this model. In our paper, we demonstrate the first attack in this arguably much more relevant setting. At the time of publication this is the only cryptanalytic attack on round-reduced version of Skein hash function.

We chose to give the simplest example in the strongest model rather than to attack the highest number of rounds. The attack we present is on a 22-round version of Skein-512 hash function. In addition to the biclique concept, an interesting feature of our attack is the application of statistical hypothesis test at the matching phase. This technique is applied for the first time and allows to cover more rounds than direct or symbolic (indirect) matching with the same computational complexity.

4.1 Second Preimage Attack on the Reduced Skein-512 Hash

Details of Skein specification are provided in Appendix A. We consider Skein-512 reduced to rounds 3–24. 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. Let f map the state after round 11 to the state before round 16. We construct a biclique of dimension 3 for f following Algorithm 1 (Section 3):

1. Define $\Delta_j^N = (0, j \lll 58, j \lll 58, 0, 0, 0, 0, 0)$ and $\nabla_i^N = (0, 0, 0, i \lll 55, i \lll 55, 0, 0, 0)$.
2. Generate Q_0 and compute $P_0, P_1, \ldots, P_7$. If the trails $0 \xrightarrow[f]{\Delta_j^N} \Delta_j$ are not based on the linear difference propagation, repeat the step.
3. Compute Q_i and check if the condition on active non-linear elements is fulfilled. If so, output a biclique.

We use a differential trail that follows a linear approximation that is a variant of the 4-round differential trail, which can be obtained in a similar way to the one presented in the paper [4]. The number of active bits is given in Table 2. Further details of the trail are provided Appendix A in Table 5. For the trails based on the 3-bit difference Δ_j^N we have 206 sufficient conditions in total. Computations of Q_i out of P_0 do not affect those conditions with probability $2^{-0.3}$ (verified experimentally). 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 most $2^{206+3} = 2^{209}$. Note that we have $1024 - 209 = 815$ degrees of freedom left.

Table 2. Number of active bits in the most dense Δ-trail in 4 rounds of Skein-512

	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	63
S^{15-P}	21	9	12	4	3	18	3	37	107 (message addition)

Probabilistic matching. The matching variable v consists of bits 30, 31, 53 of the word 1 after round 24. Due to carry effects, there is a small probability that those bits require the knowledge of the full message to be computed in both directions. This probability was experimentally estimated as 0.09. The matching bits can be computed from both chunks independently with probability 0.91, so with probability $\approx 2^{-0.1}$ we have a type-I error [22], i.e. a false positive, and the candidate is discarded (insisting on probability 1, as in earlier work, would have resulted in an attack on a smaller number of rounds).

Layout of the attack is as follows:

1. Build a biclique of dimension 3 in rounds 12-15 with key additions (key addition + 4 rounds + key addition).
2. 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.
3. Check for the match in these bits, produce 2^3 key candidates, which get reduced to $2^{2.9}$ due to the type-I error. Check them for the match on the full state.
4. Generate a new biclique out of the first one by change of key bits.
5. Repeat steps 2-5 $2^{507.5}$ times and generate $2^{507.5-509+2.9} = 2^{1.6}$ full pseudo-preimages.
6. Match one of the pseudo-preimages with the real CV_0.

Complexity. The biclique construction cost can be made negligible, since many bicliques can be produced out of one. Indeed, we are able to flip most of the bits in the message so that the biclique computation between the message injections remain unaffected, and only output states are changed. Every new biclique needs half of rounds 8-11 and 16-19 recomputing, and half of rounds 3-5 and 21-24 computing to derive the value of the matching variable. Hence each biclique tests 2^6 preimage candidates at cost of $(2+2+1.5)\cdot 8 + (2+2+2)\cdot 8 = 92$ rounds of

22-round Skein, or $2^{2.3}$ calls of the compression function, taking a recheck into account. As a result, a full pseudo-preimage is found with complexity $2^{508.4}$. We need $2^{1.6} \approx 3$ pseudo-preimages to match one of $2^{510.4}$ prefixes, so the total complexity is $2^{511.2}$.

5 Preimage Attacks on the SHA-2 Hash Functions

The SHA-2 family is the object of very intensive cryptanalysis in the world of hash functions. We briefly review parts of the specification [23] needed for the cryptanalysis in Appendix B. In contrast to its predecessors, collision attacks are no longer the major threat with the best attack on 24 rounds of the hash function [13,24]. 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 [18].

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 full layout of our attacks is provided in Table 3. The biclique is based on a 6-round trail with few conditions, easy to use as a ∇-differential. The number of attacked rounds depends significantly on its position, because:

- message injections in rounds 14-15 are partially determined by the padding rules;
- chunks do not bypass the feedforward operation due to high nonlinearity of the message schedule;
- chunks do not have maximal length, otherwise the biclique trail becomes too dense.

SHA-256. We construct a 6-round biclique with Algorithm 2, Section 3 and place it in rounds 17-22 (see Appendix C for more details of the attack).

SHA-512. The biclique is similar to the one we build for SHA-256. However, our attack on SHA-512 does not fix all the 129 padding bits of the last block. This approach still allows to generate short second 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 such as described in [16] can be used. This adds no noticeable cost as the effort 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 11 bits need to be fixed.

Details of the attack are presented in Appendix D.

Table 3. Parameters of the preimage attack on reduced SHA-2 hash functions

Target	Attack layout					
SHA-256 hash function (45-round version)	**Biclique**					
	Rounds	Dimension	Δ^M 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	23-36	37 → 38 ← 1	$A^{38}_{0,1,2,3}$	2^3	
SHA-512 hash function (50-round version)	**Biclique**					
	Rounds	Dimension	Δ^M bits:	∇^M bits:	Complexity	Freedom used
	21-26	3	$W^{21}_{60,61,62}$	$W^{26}_{53,54,55}$	2^{32}	832
	Message compensation					
	Equations			Constants used in the biclique		
	9			2		
	Chunks		**Matching**			
	Forward	Backward	Partial matching	Matching bits	Complexity per match	
	6-20	27-40	41 → 43 ← 5	$A^{43}_{0,1,2}$	2^3	

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. Control over the tweak value gives us additional freedom both in chunks and the biclique construction.

The attack parameters are listed in Table 4. We build a biclique in rounds 16-23, and apply the attack to rounds 2-38. Bicliques are constructed by Algorithm 3 (Section 3). We use two differential trails: based on Δ^M (Δ-trail) for rounds 16-19 (including key addition in round 19) and based on ∇^M (∇-trail) for rounds 20-23. The Δ- and ∇- trails are based on the evolution of a single difference in the linearized Skein and have probability 2^{-52} and 2^{-29} respectively.

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 experimentally). 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.

Table 4. Parameters of the preimage attack on the 37-round Skein-512 compression function

Biclique					
Rounds	Dimension	Δ^M bits	∇^M 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	$32 \rightarrow 38 = 2 \leftarrow 7$	I_{25}^{3}	2^2	$2^{1.1}$

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. The number of rounds we obtain for the compression function setting is in both cases comparable to [18], the latter however does not allow extension to the hash function nor does it violate any “traditional” security requirement. 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 Δ_1^M and ∇_1^M. As a result, we get sparse trails with few conditions, and can extend them to 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 Δ_1^M to W^{17}, and assume the linear evolution of Δ_1^M 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 introduced a new concept of bicliques for meet-in-the-middle attacks. We presented several applications of biclique cryptanalysis, including the best preimage attacks so far on SHA-256, SHA-512, and the SHA-3 finalist Skein. In line with most cryptanalytic work, we focused on obtaining results on as many rounds as possible. Though all the functions in this paper are ARX-based, our technique can also be applied to other narrow-pipe designs.

Overall, the differential view gives a cryptanalyst much more freedom and flexibility compared to previous attacks. We can outline the following benefits of applying the biclique concept:

- 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 [6,15,21,28], condition propagation [9], and rebound techniques [20].

Status of SHA-2 and Skein-512. 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. Based on these results we conclude that Skein-512 is more resistant against splice-and-cut cryptanalysis than SHA-512.

Other Applications of Biclique Cryptanalysis. Soon after the initial circulation of this work, the idea of biclique cryptanalysis found other applications. Bicliques have large potential in attacks on block ciphers, as has been demonstrated by recent attacks on the full versions of popular block ciphers. Among them we mention key recovery faster than brute force for AES-128, AES-192, and AES-256 by Bogdanov et al. [8]. Cryptanalysis of AES employed algorithms for biclique construction which are partly covered in Section 3. In this context we also mention new and improved results on Kasumi by Jia et al. [14] and IDEA by Biham et al. [7] as well as more results announced both publicly [12,19,30] and privately.

Acknowledgements. Part of this work was done while Christian Rechberger was with KU Leuven and visiting MSR Redmond, and while Alexandra Savelieva was visiting MSR Redmond. The authors would like to thank Eik List and anonymous reviewers for useful comments on earlier versions of the paper.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for Step-Reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009)

2. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
4. Aumasson, J.-P., Çalık, Ç., Meier, W., Özen, O., Phan, R.C.-W., Varıcı, K.: Improved Cryptanalysis of Skein. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 542–559. Springer, Heidelberg (2009)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: RadioGatun, a belt-and-mill hash function. In: NIST Cryptographic Hash Workshop (2006), `http://radiogatun.noekeon.org/`
6. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
7. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New Data-Efficient Attacks on Reduced-Round IDEA. Cryptology ePrint Archive, Report 2011/417 (2011), `http://eprint.iacr.org/`
8. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011), `http://eprint.iacr.org/2011/449`
9. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
10. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family, version 1.3 (October 1, 2010)
11. 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 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
12. Hong, D.: Biclique attack on the full HIGHT. To appear in ICISC 2011 (2011)
13. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and Other Nonrandom Properties for Step-Reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 276–293. Springer, Heidelberg (2009)
14. Jia, K., Yu, H., Wang, X.: A meet-in-the-middle attack on the full KASUMI. Cryptology ePrint Archive, Report 2011/466 (2011), `http://eprint.iacr.org/`
15. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
16. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
17. Khovratovich, D., Nikolić, I., Rechberger, C.: Rotational Rebound Attacks on Reduced Skein. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 1–19. Springer, Heidelberg (2010)
18. Lamberger, M., Mendel, F.: Higher-order differential attack on reduced SHA-256 (2011), `http://eprint.iacr.org/2011/037.pdf`
19. Mala, H.: Biclique cryptanalysis of the block cipher SQUARE. Cryptology ePrint Archive, Report 2011/500 (2011), `http://eprint.iacr.org/`

20. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
21. Naito, Y., Sasaki, Y., Shimoyama, T., Yajima, J., Kunihiro, N., Ohta, K.: Improved Collision Search for SHA-0. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 21–36. Springer, Heidelberg (2006)
22. Neyman, J., Pearson, E.S.: The testing of statistical hypotheses in relation to probabilities a priori. In: Proc. Camb. Phil. Soc. (1933)
23. NIST. FIPS-180-2: Secure Hash Standard (August 2002), http://www.itl.nist.gov/fipspubs/
24. Sanadhya, S.K., Sarkar, P.: New Collision Attacks against Up to 24-Step SHA-2. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008)
25. Sasaki, Y., Aoki, K.: Preimage Attacks on Step-Reduced MD5. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 282–296. Springer, Heidelberg (2008)
26. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
27. Su, B., Wu, W., Wu, S., Dong, L.: Near-Collisions on the Reduced-Round Compression Functions of Skein and BLAKE. Cryptology ePrint Archive, Report 2010/355 (2010), http://eprint.iacr.org/
28. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
29. Yu, H., Chen, J., Jia, K., Wang, X.: Near-Collision Attack on the Step-Reduced Compression Function of Skein-256. Cryptology ePrint Archive, Report 2011/148 (2011), http://eprint.iacr.org/
30. Chen, S.Z., Xu, T.M.: Biclique Attack of the Full ARIA-256. Cryptology ePrint Archive, Report 2012/011 (2012), http://eprint.iacr.org/

A Skein Specification and Details of Differential Trail Design

Skein-512 is based on the block cipher Threefish-512 — a 512-bit block cipher with a 512-bit key parametrized by a 128-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 parameters of message block M.

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

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

The subkey $K^s = (K^s_0, K^s_1, \dots, K^s_7)$ is produced out of the key $K = (K[0], K[1], \dots, K[7])$ as follows:

$$K^s_j = K[(s+j) \bmod 9], \quad 0 \le j \le 4; \quad K^s_5 = K[(s+5) \bmod 9] + T[s \bmod 3];$$
$$K^s_6 = K[(s+6) \bmod 9] + T[(s+1) \bmod 3]; \quad K^s_7 = K[(s+7) \bmod 9] + s,$$

where the additions are all modulo 2^{64}, 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 constant C_{240} optimized against rotation attacks.

One round transforms the internal state as follows. Eight words $I^0, I^1, \dots, I^7$ 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. Details of these operations are irrelevant for the high-level description and can be found in [10]. We use the following notation for the internal states in round r:

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

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 with the combination $\Delta K[6] = \Delta K[7] = \Delta T[1] = \delta$, which gives difference $(0, 0, \dots, 0, \delta)$ in the subkey K^0 and $(\delta, 0, 0, \dots, 0)$ 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:

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

B Specification of the SHA-2 Family of Hash Functions

The SHA-2 hash functions are based on a compression function that updates the state of eight 32-bit state variables A, ..., H according to the values of 16 32-bit words M_0, ..., 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):

$$W^i = \begin{cases} M^i & \text{for } 0 \le i < 15 \\ \sigma_1(W^{i-2}) + W^{i-7} + \sigma_0(W^{i-15}) + W^{i-16} & \text{for } 15 \le i < n \end{cases} . \tag{11}$$

Table 5. Details of the most dense Δ-trail for the result on the reduced Skein-512 hash function

Round	Active bits
Before Round 12	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0001110000000000000000000000000000000000000000000000000000000000
After Round 12	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0001110000000000000000000000000000000000000111000000000000000000 0001110000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
After Round 13	0001110000000000000000000000000000000000000111000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0001110000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0001110000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0001110000000000011100000000000000000000000111000000000001110000
After Round 14	0001110000000000000000000000000000000000000000000000000000000000 0001110000000000000000000000000000000000000111000000000000000000 0001110000000000000000000000000000000000000000000000000000000000 1111110000000011111100001110000000000011100111000000000001110000 0001110000000000011100000000000000000000000111000000000001110000 0001110000000000000000000000111000000000000000000000000000000000 0001110000000000000000000000000000000000000111000000000000000000 0001110000000000000000000000000000000000000000000000000000000000
After Round 15	1110000000000011111100001110000000000011100111000000000001110000 0000000000000000000000000000000000011100000111000000000000011100 0000000000000000011100000000111000000000000111000000000001110000 0000000000000000000000000000000000000000000110110000000000000000 0000000000000000000000000000000000000000000111000000000000000000 0000000000011100011100000000111000011100000111000000000001110000 0000000000000000000000000000000000000000000111000000000000000000 1111110011100011111100110110011111100011100000111000011101110000

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

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

Here K^i is a round constant. The bitwise boolean functions Maj and Ch as well as two GF(2)-linear functions $\Sigma_0(x)$ and $\Sigma_1(x)$ used in the round function, and message schedule functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined in Table 6. $A \ggg x$ and $A \gg x$ denote bit-rotation and bit-shift of A by x positions to the right.

C Details on the 46-Round SHA-256 Attack

C.1 Biclique Construction

Here we provide more details on the biclique construction algorithm:

Table 6. Details of SHA-2 hash family internal operation

Function	**SHA-2 Family**	
	SHA-224 and SHA-256	SHA-384 and SHA-512
$\mathrm{Ch}(x,y,z)$	$x \wedge y \oplus \overline{x} \wedge z$	
$\mathrm{Maj}(x,y,z)$	$x \wedge y \oplus x \wedge z \oplus y \wedge z$	
$\Sigma_0(x)$	$(x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22)$	$(x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39)$
$\Sigma_1(x)$	$(x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25)$	$(x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41)$
$\sigma_0(x)$	$(x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$	$(x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7)$
$\sigma_1(x)$	$(x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$	$(x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6)$

1. Fix a group of 6-round differential trails $\nabla_i \xrightarrow{\nabla_i^M} 0$ (the one based on 3-bit difference is listed in Table 7). Derive a set of sufficient conditions on the internal states (Table 8).
2. Fix the message compensation equations with constants $c_1, c_2, \ldots, c_9$ (Section C.2).
3. Fix an arbitrary Q_0 and modify it so that most of conditions in the computation $Q_0 \to P_0$ are fulfilled. Derive Q_i out of Q_0 by applying ∇_i.
4. Fix a group of 2-round trails (the one based on 3-bit difference is given in Table 7) ($\Delta W^{17} \to \Delta S^{19}$) as a Δ-trail (Equation (6)) 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 $Q_0 \to P_j, j = 0, \ldots, 7$ are fulfilled. Produce all P_j.

Finally, we produce $Q_0, \ldots, Q_7$ and $P_0, \ldots, P_7$ that conform to the biclique equations.

The complexity of building a single biclique is estimated as 2^{32}. 7 message words left undefined in the message compensation equations give 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 producing 2^2 matches on 4 bits is 2^3 calls of the compression function. Since we need 2^{252} matches, the complexity of the pseudo-preimage search is 2^{253}. 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 $\approx 2^{1.5} \times 24$ words.

C.2 Message Compensation

Since any consecutive 16 message words in SHA-2 bijectively determine the rest of the message block 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 discovered that with W^{17} and W^{22} selected as the words with for a 6-step initial structure, 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):

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

$W^{14}, \ldots, W^{16}, W^{18}$, 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. We use bits 25, 26, 27 as neutral in W_{17}. To prevent this difference from interleaving with the backward trail difference in round 19, we restrict the behavior of the forward trail as specified in Table 7 (aggregated conditions are given in Table 8).

C.3 Trails

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 8.

Table 7. Details for biclique in SHA-256. Differential ∇- and Δ- trails (active bits).

$\Lambda' = \{6, 11, 12, 16, 17, 20, 23, 24, 29, 30\}, \Phi = \Sigma_1\{25, 26, 27\} = \{0, 1, 2, 14, 15, 16, 19, 20, 21\}$, $*$ refers to an arbitrary difference.

Trail	R-nd	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	
Δ	18	*	-	-	-	25,26,27	-	-	-	-	2
Δ	19	*	*	-	-	Φ	25,26,27	-	-	-	5,6

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. Next, in round 18 we check whether the value of E stops carries in the forward trail. If not, we

Table 8. Sufficient conditions for the Δ- and ∇-trails in SHA-256

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.
A^i – i-th bit of A; $\Lambda = \Sigma_1\{22, 23, 31\} = \{6, 11, 12, 16, 17, 20, 25, 29, 30\}$

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

change the value of D in the starting state accordingly. Then we sequentially modify the initial state 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. 12 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. 12. We spend $256 + 5 \cdot 32 = 416$ degrees of freedom for the biclique fulfilling as few as 47+42 (Table 8) conditions. After the biclique is fixed, there are $768 - 416 = 352$ degrees of freedom left. We spend $32 + 32 + 2 = 66$ for the padding, leaving 286 degrees of freedom. Therefore, one biclique is enough for the full attack.

D Details on the 50-Round SHA-512 Attack

D.1 Biclique Construction

Steps of the algorithm are similar to those in Section C.1, except the trails are described in Table 9. By applying similar reasoning, we estimated that a full preimage can be found with complexity $\approx 2^{511.5}$ and memory $\approx 2^{1.5} \times 24$ words.

D.2 Message Compensation

The system of compensation equations is defined as follows:

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

We use 1 LSB of W^{13} and 10 LSB of W^{15} for padding. The choice of constants c_8, c_9 and fixed lower 53 bits of W^{26} provide us with sufficient freedom. 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. 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 lower 33 bits of W^{26} and c_9, and lower 25 bits of c_8.

D.3 Trails

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}. We also use bits 60, 61, 62 as neutral in W^{21}. To prevent this difference from interleaving with the backward trail difference in round 19, we restrict the behavior of the forward trail. The trails are depicted in Table 9.

Table 9. Details for biclique in SHA-512. Differential ∇- and Δ-trails (active bits).

$\Lambda = \Sigma_1\{53, 54, 55\} = \{12, 13, 14, 35, 36, 37, 39, 40, 41\}, \Phi = \Sigma_1\{60, 61, 62\} = \{17, 20, 21, 42, 43, 44, 46, 47, 48\}$,
$*$ refers to an arbitrary difference.

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