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

Dmitry Khovratovich[1] and 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 the 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 demonstrate how our concept results in the first cryptanalysis of the 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

The last years saw an exciting progress in preimage attacks on hash functions. In contrast to collision search [29, 26], where differential cryptanalysis has been in use since 90s, there had been no similarly powerful tool for preimages. The situation changed dramatically in 2008, when the so-called *splice-and-cut* framework has been applied to MD4 and MD5 [2, 24], and later to SHA-0/1/2 [1, 3], Tiger [10], and other primitives. Despite amazing results on MD5, the applications to SHA-x primitives seemed to be limited by the internal properties of the message schedule procedures. The only promising element, the so-called *initial structure* tool, remained very informal and complicated. Recent results on preimage attacks and, maybe surprisingly, on key recovery attacks on block ciphers, have demonstrated that these obstacles can be mitigated with the help of a new concept called *bicliques*.

This paper presents the first work on the concept of biclique cryptanalysis. Originally, the new concept was a tool for preimage attacks only. However, it quickly became crucial for new key recovery attacks on block ciphers as well as we mention at the end of the introduction. In this paper we concentrate on the hash function setting alone, and focus on new definitions and algorithms. As applications, we present an attack on the Skein hash function (the only one existing so far) and briefly describe attacks on SHA-2 hash functions.

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, 24] that compression functions with permutation-based message schedule are vulnerable to this kind of attack as chunks can be 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 [25, 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 the initial structure 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	
Section 4	Skein-512	22	2^{508}	2^{511}	2^6
Section 4	Skein-512	72	-	$2^{511.76}$	negl.
Section 6	Skein-512	37	$2^{511.2}$	-	2^{64}
[1, 10]	SHA-256	43	$2^{251.9}$	$2^{254.9}$	2^6
Section 5	SHA-256	45	2^{253}	$2^{255.5}$	2^6
Section 6	SHA-256	52	2^{255}	-	2^6
[1, 10]	SHA-512	46	2^{509}	$2^{511.5}$	2^6
Section 5	SHA-512	50	2^{509}	$2^{511.5}$	2^4
Section 6	SHA-512	57	2^{511}	-	2^6

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

Other applications of biclique cryptanalysis

Soon after the initial circulation of this work, the idea of biclique cryptanalysis found other applications. 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. [13] and IDEA by Biham et al. [7] as well as more results announced both publicly [11, 18] and privately.

2 Bicliques

In this section we introduce splice-and-cut preimage attacks with bicliques. We consider the most popular Davies-Meyer mode: $H = E_M(CV) \oplus CV$, where CV is the chaining variable, and E is the block cipher keyed with the message M.

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

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

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

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

An adversary selects a variable v outside of f (w.l.o.g. between P_j and H) and checks if

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

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

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:

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

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 messages.

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.

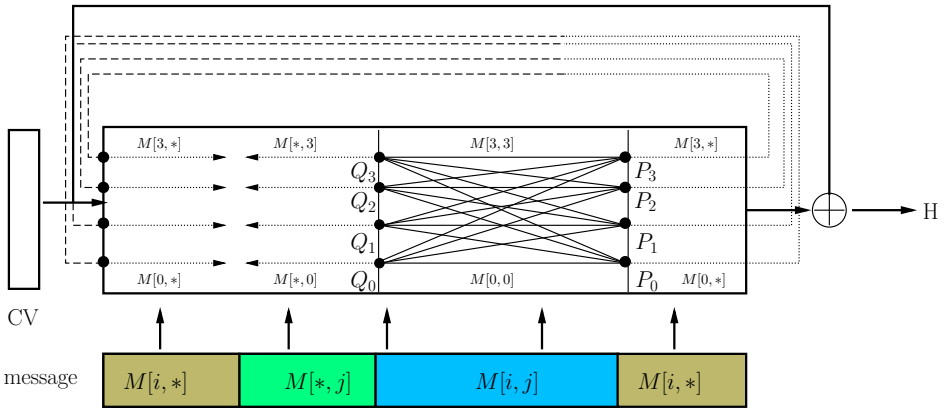

Fig. 1. Biclique of dimension 2 in the meet-in-the-middle attack.

For most algorithms we adopt a differential view on bicliques as it allows for numerous tools from differential cryptanalysis to be employed. Consider a single mapping in Equation (1)

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

It is also called a *basic computation*. Consider the other mappings as differentials to the basic computation:

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

so that

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

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

In the following algorithms we show how to reduce the number of differentials needed for a biclique, and hence construct a biclique efficiently.

Algorithm 1. Let the differences in a message group $\mathcal{M}$ be defined as the following linear function:

$$\Delta^M_{i,j} = \Delta^M_j \oplus \nabla^M_i \tag{4}$$

Let us fix Q_0 and construct P_j as follows:

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

As a result, we get a set of trails:

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

Let us also construct Q_i out of P_0:

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

and get another set of trails:

$$\nabla_i \xrightarrow[f]{\nabla^M_i} 0. \tag{8}$$

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:

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

To estimate the complexity, assume that the computation (7) does not affect active non-linear elements in the trails (6) with probability 2^{-t}. Then the probability that 2^d such computations affect no condition is 2^{-t2^d}. Therefore, Equation (9) is satisfied with probability 2^{-t2^d}, so we need 2^{t2^d} solutions to Equation (6) to build a biclique (which is feasible for small d). This approach is used in the preimage attack on the hash function Skein-512.

For non-ARX primitives with predictable diffusion this algorithm can be easily made deterministic. For example, it is easy to construct the truncated differential trails for AES [8] and Square that do not share active non-linear components with probability 1 (Figure 2). As a result, an attack algorithm can be simply explained at a picture of trails.

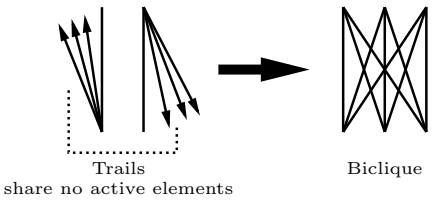

Fig. 2. Biclique out of non-interleaving trails.

Algorithm 2. (Modification of Algorithm 1 for the case when we can control internal state and message injections within the biclique). Assume that the mapping f uses several independent parts (blocks) of message M via the message injections (like in SHA-2). Consider a message group with property (4) but do not define the messages yet. Fix a state Q_0 and sets of trails (6) and (8) that do not share active non-linear components. Then find a message $M[0,0]$ such that the computation

$$Q_0 \xrightarrow[f]{M[0,0]} P_0$$

conforms to both sets of trails. Since the sets do not share active non-linear components, we get

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

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

Since we control message injections in f, we are able to define $M[0,0]$ block by block similarly to the trail backtracking approach [5]. If the message schedule is non-linear, the differential trails (6) and (8) may depend on $M[0,0]$. Furthermore, parts of message $M[0,0]$ may remain undefined as they are not used in f. A procedure that ensures that the message $M[0,0]$ is well-defined, and the trails (6) and (8) do not contradict, is called *message compensation.*

Algorithm 3. (for bicliques of dimension 1) We apply this rebound-style 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:

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

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 for a given message M and the quartet of states is the quartet in the middle of the boomerang attack, which happens with probability p^2q^2 for a random M under some independence assumptions. Then we derive input states Q_0, Q_1 and output states P_0, P_1, which are linked as follows:

$$\begin{gathered}
Q_0 \xrightarrow[f_1]{M} S \xrightarrow[f_2]{M} P_0; \\
Q_0 \xrightarrow[f_1]{M \oplus \Delta^M} S \oplus \Delta \xrightarrow[f_2]{M \oplus \Delta^M} P_1; \\
Q_1 \xrightarrow[f_1]{M \oplus \nabla^M} S \oplus \nabla \xrightarrow[f_2]{M \oplus \nabla^M} P_0; \\
Q_1 \xrightarrow[f_1]{M \oplus \Delta^M \oplus \nabla^M} S \oplus \Delta \oplus \nabla \xrightarrow[f_2]{M \oplus \Delta^M \oplus \nabla^M} P_1.
\end{gathered}$$

Therefore, we get a biclique, where the message group is defined as follows:

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

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 (Figure 3).

This algorithm is applied in the preimage attack on the Skein compression function. It also has been used in so-called long biclique attacks on AES.

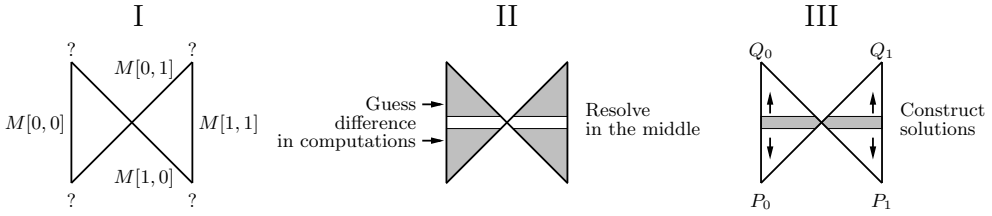

Fig. 3. Rebound-style algorithm for biclique construction.

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

Skein [9] is a SHA-3 finalist, and hence gets a lot of cryptanalytic attention. Differential [4] and rotational cryptanalysis [16] led the authors of Skein to tweak the design twice. As a result, a rotational property, which allowed cryptanalyst to penetrate the highest number of rounds, 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, 27]. Very recently near-collisions attacks on up to 32 rounds of Skein-256 were demonstrated [30].

The cryptanalysis of the Skein hash function, however, is very limited, and since the first publication of this work there has been no advance in this direction. 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). In addition to using the biclique concept, on of the interesting features of our attack is that we, apparently for the first time, utilize a statistical hypothesis test to improve the matching phase instead of a direct or a symbolic (indirect) matching. Without it, less rounds could be covered with basically the same computational complexity.

4.1 Description of Skein-512

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 several parameters including the index of the last bit of the message.

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

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

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:

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

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 constant C_{240} optimized against rotation attacks.

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 B. 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 the reduced Skein-512

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 $\Delta_j^M = (0, j \ll 58, j \ll 58, 0, 0, 0, 0, 0)$ and $\nabla_i^M = (0, 0, 0, i \ll 55, i \ll 55, 0, 0, 0)$.
2. Generate Q_0 and compute $P_0, P_1, \ldots, P_7$. If the trails $0 \xrightarrow[f]{\Delta_j^M} \Delta_j$ 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 from the paper [4], which has probability 2^{-68} and, thus, 68 bit conditions if taking the message addition into account. For the trails based on the 3-bit difference Δ_j^M we have, due to some overlaps, only 197 sufficient conditions in total. A computation of Q_i out of P_0 do not affect those conditions with probability $2^{-0.3}$ (checked on a PC), or 2^{-3} in total. 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 $2^{197+3} = 2^{200}$. Note that we have $1024 - 200 = 824$ degrees of freedom left.

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 has been computed experimentally and equals 0.09. Therefore, a matching pair of computations yields a pseudo-preimage with probability $2^{-509.1}$, and we need to use $2^{506.1}$ bicliques for this purpose.

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.

On step 3. We have checked experimentally that the matching bits can be computed from both chunks independently with probability 0.91, so with probability $2^{-0.1}$ we have a type-I error [21], i.e. a false positive, and the candidate is discarded. Insisting on probability 1, as done in earlier work, would have lead to a redesign of the attack for a smaller number of rounds.

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 $3 = 2^{1.6}$ pseudo-preimages to match one of $2^{510.4}$ prefixes, so the total complexity is $2^{511.2}$.

4.3 Second preimage search for full Skein-512

Skein-512 appears to be the only hash function in the finalist selection of the SHA-3 competition that allows for time-complexity gains over brute-force search using cryptanalytic meet-in-the-middle strategies[1]. In here we explore this further.

We start with the simple observation that when looking through a message space the first rounds need only be partially computed. When using the 64-bit modular addition is the cost metric, the first round can be for free, the second rounds needs only one instead of 8 computations, etc, saving more than 3 round computations in total. Likewise, if only a few output bits instead of all need to be computed, e.g. for matching with the target hash value, a similar number of computations can be saved. Similar observations can and have been made for other primitives, however as Skein-512 is the only narrow-pipe SHA-3 finalist, we can go further. Those bits on which the check is performed need not be at the end of the compression function call, but can be at any point in the internal state. This allows to also have savings in another chunk. One simple example of such savings are neutral bit effects. Similar to [16], in experiments we found that for 5 rounds in the backwards direction, many neutrals bits to not affect a number of state bits with probability 1. These neutral bits can in turn be used as the inner loop of the search space. The total number of rounds that are saved are hence at least 11, leading to a preimage search complexity of no more than $2^{511.76}$. By spending some computation (that is later amortized) to find suitable chaining values that allow for longer neutral bits, or by simply choosing a suitable CV in the compression function setting, these results can be improved further.

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. 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, 23]. 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 [17]. 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 [17], the later however does not allow extension to the hash function nor does it violate any "traditional" security requirement.

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.

SHA-256 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 4. The biclique is constructed with Algorithm 2, Section 3.

[1] The narrow pipe Skein-256-256 also allows for the approach, but all versions of Keccak, Grøstl, and JH are wide-pipe and do not allow this. For Blake the situation is less clear.

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 [15] 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.

The biclique is constructed by an algorithm similar to the attack on SHA-256 (Algorithm 2, Section 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. In the compression function setting we control the tweak value, which gives us additional freedom both in chunks and the construction of the biclique.

The attack parameters are listed in Table 3 in the Appendix. We build a biclique in rounds 24-31, and apply the attack to rounds 2-38, i.e., to the 37-round compression function.

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 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 Δ_1^M and ∇_1^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 5 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 reconsidered meet-in-the-middle attacks and introduced a new concept of bicliques. Bicliques have large potential in attacks on narrow-pipe hash functions and block ciphers, as has been demonstrated by recent attacks on the full versions of popular block ciphers.

To emphasize new ideas and methods behind the new concept, we focused on clear definitions and a variety of construction algorithms. As for applications, in the main text we described basic steps in the best attacks so far on SHA-256, SHA-512, and the SHA-3 finalist Skein, with more details left for the Appendix.

As for the employed techniques, we additionally benefit from the following features:

- 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, 14, 20, 28], and rebound techniques [19];
- Utilizing a statistical test for matching, instead of a direct or symbolic matching.

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.

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. 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.

For Skein-512 we also apply the meet-in-the-middle approach to obtain a computational-complexity gain for its full 72-round version. While we don't claim this to be an attack, it seems worthwhile to point out that this is the only hash function in the SHA-3 finalist selection that allows for such an approach.

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. 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'04*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
7. Eli Biham, Orr Dunkelman, Nathan Keller, and Adi Shamir. New Data-Efficient Attacks on Reduced-Round IDEA. Cryptology ePrint Archive, Report 2011/417, 2011. `http://eprint.iacr.org/`.
8. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. Cryptology ePrint Archive, Report 2011/449, 2011. `http://eprint.iacr.org/2011/449`, to appear in ASIACRYPT 2011.
9. 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).
10. 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.
11. Deukjo Hong. Biclique attack on the full HIGHT, 2011. to appear in ICISC 2011.
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. Keting Jia, Honbo Yu, and Xiaoyun Wang. A meet-in-the-middle attack on the full KASUMI. Cryptology ePrint Archive, Report 2011/466, 2011. `http://eprint.iacr.org/`.
14. Antoine Joux and Thomas Peyrin. Hash functions and the (amplified) boomerang attack. In *CRYPTO'07*, volume 4622 of *LNCS*, pages 244–263. Springer, 2007.
15. 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.
16. 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.
17. Mario Lamberger and Florian Mendel. Higher-order differential attack on reduced SHA-256. available at `http://eprint.iacr.org/2011/037.pdf`, 2011.
18. Hamid Mala. Biclique cryptanalysis of the block cipher SQUARE. Cryptology ePrint Archive, Report 2011/500, 2011. `http://eprint.iacr.org/`.
19. 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.
20. 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.
21. J. Neyman and E.S Pearson. The testing of statistical hypotheses in relation to probabilities a priori. *Proc. Camb. Phil. Soc.*, 1933.
22. NIST. FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.
23. 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.
24. Yu Sasaki and Kazumaro Aoki. Preimage attacks on step-reduced MD5. In *ACISP'08*, volume 5107 of *LNCS*, pages 282–296. Springer, 2008.

25. 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.
26. 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.
27. 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/`.
28. 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.
29. 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.
30. Hongbo Yu, Jiazhe Chen, Ketingjia, and Xiaoyun Wang. Near-Collision Attack on the Step-Reduced Compression Function of Skein-256. Cryptology ePrint Archive, Report 2011/148, 2011. `http://eprint.iacr.org/`.

A Specification of SHA-2 Family of Hash Functions

We briefly review parts of the specification [22] 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):

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

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 C.

B 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:

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

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 [9].

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:

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

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

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:

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

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	$32 \rightarrow 39 = 2 \leftarrow 7$	I^3_{25}	2^2	$2^{1.1}$

Table 3. Parameters of the preimage attack on the Skein-512 compression function

C More details on SHA-2 specification

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

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

Here K^i is a round constant.The 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:

$$\mathrm{Ch}(x, y, z) = x \wedge y \oplus \overline{x} \wedge z \tag{12}$$

$$\mathrm{Maj}(x, y, z) = x \wedge y \oplus x \wedge z \oplus y \wedge z \tag{13}$$

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:

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

and for SHA-384 and SHA-512:

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

D Details on the 46-round SHA-256 attack

D.1 Biclique construction

Here we provide more details on the biclique construction algorithm:

1. Fix a group of 6-round differential trails (the one based on 3-bit difference is listed in Table 5)
$$\nabla_i \xrightarrow{\nabla_i^M} 0.$$
Derive the set of sufficient conditions on the internal states (Table 7).
2. Fix the message compensation equations with constants $c_1, c_2, \ldots, c_9$ (Section D.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 6) ($\Delta W^{17} \to \Delta S^{19}$) 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.

An algorithm for the biclique is detailed in Appendix, Section D.3. Finally, we produce $Q_0, \ldots, Q_7$ and $P_0, \ldots, P_7$ that conform to the biclique equations.

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	23-37	$37 \to 38 \leftarrow 1$	$A^{38}_{0,1,2,3}$	2^3	

Table 4. Parameters of the preimage attack on the 45-round SHA-256

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.

D.2 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):

$$\begin{array}{lll} -\sigma_1(W^{25}) + W^{27} = c_1; & -W^{19} - \sigma_1(W^{24}) + W^{26} = c_2 & -\sigma_1(W^{23}) + W^{25} = c_3 \\ -\boxed{W^{17}} + W^{24} = c_4 & -\sigma_1(W^{21}) + W^{23} = c_5; & -\sigma_1(W^{19}) + W^{21} = c_6 \\ -\sigma_1(\boxed{W^{17}}) + W^{19} = c_7; & W^{12} + \sigma_0(W^{13}) = c_8; & W^{13} + \boxed{W^{22}} = c_9 \end{array} \tag{14}$$

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.

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 22, 23, and/or 31 in W_{22}. The trail is briefly depicted in Table 5 with references to the sufficient conditions (which work out for all the 7 possible differences) in Table 7.

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 6. The aggregated conditions, which make each forward trail keep the backward ones unaffected, are given in Table 7.

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

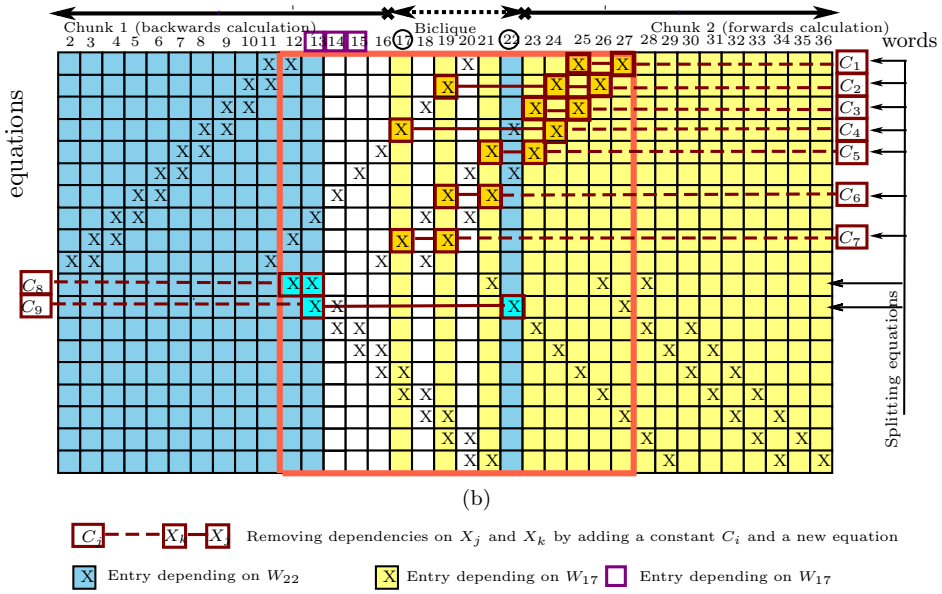

Fig. 4. Message dependencies: (a) before and (b) after message compensation in SHA-256

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

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. 14 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. 14. Therefore, we spend $256 + 5 \cdot 32 = 416$ degrees of freedom for the biclique fulfilling as few as $47 + 42$ (Table 7) 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	A	B	C	D	E	F	G	H	Cond-s
18	*	-	-	-	25,26,27	-	-	-	2
19	*	*	-	-	Φ	25,26,27	-	-	5,6

Table 6. 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.

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 7. 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.
$\Lambda = \Sigma_1\{22, 23, 31\} = \{6, 11, 12, 16, 17, 20, 25, 29, 30\}$

E Details on the 50-round SHA-512 attack

E.1 Biclique construction

Attack layout The basic parameters of the pseudo-preimage attack are given in Table 8. More details:

1. Fix a group of 6-round differential trails (Table 9) for the differential
$$\nabla_i \xrightarrow{\nabla^M} 0.$$
Derive the set of sufficient conditions on the internal states.
2. Fix the message compensation equations with 9 constants (Appendix E.2).
3. Fix an arbitrary Q_0 and modify it so that the 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 3-round trails (Table 10) ($\Delta W^{21} \to \Delta S^{23}$) as Δ-trails (Equation (6)) 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 $Q_0 \to P_j, j = 0, \ldots, 7$ are fulfilled. Produce all P_j.

Trail details for the biclique are detailed further. 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 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. Since

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}	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 8. Parameters of the preimage attack on the 50-round SHA-512

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.

E.2 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}.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.

E.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}. The trail is depicted in Table 9 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 10. 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.

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.

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 9. 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 10. 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.

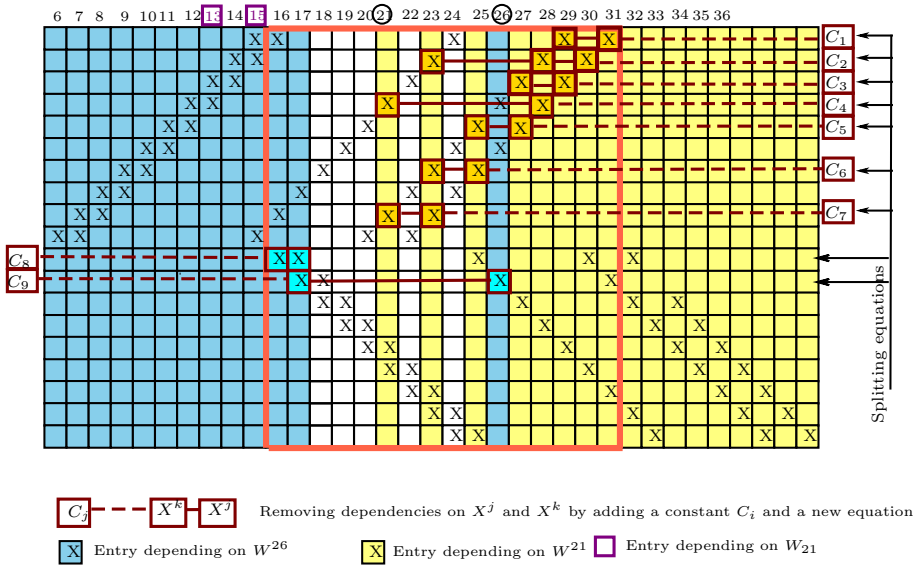

Fig. 5. Message compensation in SHA-512