Efficient Construction of the Kite Generator Revisited

Orr Dunkelman[1][0000−0001−5799−2635] and Ariel Weizman[2][0000−0001−7177−0473]

[1] Computer Science Department, University of Haifa, Israel
[2] Department of Department of Mathematics, Bar-Ilan University, Israel

Abstract. The *kite generator*, first introduced by Andreeva et al. [1], is a strongly connected directed graph that allows creating a message of almost any desired length, connecting two chaining values covered by the *kite generator*. The *kite generator* can be used in second pre-image attacks against (dithered) Merkle-Damgård hash functions.
In this work we discuss the complexity of constructing the *kite generator*. We show that the analysis of the construction of the *kite generator* first described by Andreeva et al. is somewhat inaccurate and discuss its actual complexity. We follow with presenting a new method for a more efficient construction of the *kite generator*, cutting the running time of the preprocessing by half (compared with the original claims of Andreeva et al. or by a linear factor compared to corrected analysis). Finally, we adapt the new method to the dithered Merkle-Damgård structure.

1 Introduction

One of the important fundamental primitives in cryptography is cryptographic hash functions. They are widely used in digital signatures, hashing passwords, message authentication code (MAC), etc. Hence, their security has a large impact on the security of many protocols.

Up until the SHA3 competition, the most widely used hash function construction was the Merkle-Damgård one [5, 11]. The Merkle-Damgård structure extends a compression function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ into a hash function $\mathcal{MDH}^f : \{0,1\}^* \to \{0,1\}^n$. Indeed, the Merkle-Damgård structure is still widespread, as can be seen from the wide use of the SHA2 family [12]. However, in the last fifteen years, a series of works pointed out several structural weaknesses in the Merkle-Damgård construction and its dithered variant [1, 6–9].

One way for comparing between different structures of cryptographic hash functions is considering generic attacks. Naturally, generic attacks use complex algorithms and data structures, and often become used as subroutines in other attacks. In such cases, the accurate analysis of these algorithms and data structures becomes very important. For example, Kelsey and Kohno suggest a special data structure, called the *diamond structure*, which is a complete binary tree with 2^ℓ leaves, to support the *herding* attack [8]. Blackburn et al. [4] point out an inaccuracy in Kelsey-Kohno's analysis and fix it, resulting in an increased

time complexity. Later work presented new algorithms for more efficient constructions of the diamond structure [10, 14].

A different second pre-image attack is based on the *kite generator*. This is a long message (with 2^k blocks) second pre-image attack due to Andreeva et al. [1] on the Merkle-Damgård structure and its dithered variant [13]. The *kite generator* is a strongly connected directed graph of 2^{n-k} chaining values that for each two chaining values a_1, a_2 covered by the *kite generator*, there exist a sequence of message blocks of almost any desired length that connects a_1 to a_2. Their analysis claims that the *kite generator*'s construction takes about 2^{n+1} compression function calls.

We start this paper by pointing out an inaccuracy in their construction based on some theorems from the Galton-Watson branching process field: We show that the resulting graph, using the original construction, is not strongly connected and therefore is unusable in the online phase. We proceed by offering corrected analysis that shows that the construction of the kite generator takes about $(n-k) \cdot 2^n$ compression function calls.

We then show a completely different method to build the kite generator. This new method allows constructing the kite generator in time of 2^n compression function calls, i.e., it takes half the time of the originally inaccurate claim. Finally, we adapt all these issues to the dithered variant of the Merkle-Damgård structure.

This paper is organized as follows: Section 2 gives notations and definitions used in this paper. In Section 3 we quickly recall Andreeva et al.'s second pre-image attack, and most importantly, the construction of the *kite generator*. We identify and analyze the real complexity of constructing a usable *kite generator* in Section 4. We introduce a new method for constructing kite generators in Section 5. We treat the analysis of the *kite generator* and the new construction in the case of dithered Merkle-Damgård in Section 6. Finally, we conclude the paper in Section 7.

2 Notations and Definitions

Definition 1. *A cryptographic hash function is a function $H : \{0,1\}^* \to \{0,1\}^n$, that takes an arbitrary length input and transforms it to an n-bit output such that $H(x)$ can be computed efficiently, while the function has three main security properties:*

1. Collisions resistance: It is hard to find (with high probability) an adversary that could produce two different messages M, M' such that $H(M) = H(M')$ in less than $\mathcal{O}(2^{n/2})$ calls to $H(\cdot)$.
2. Second pre-image resistance: Given M such that $H(M) = h$, an adversary cannot produce (with high probability) an additional message $M' \neq M$ such that $H(M') = h$ in less than $\mathcal{O}(2^n)$ calls to $H(\cdot)$.
3. Pre-image resistance: Given a hash value h, an adversary cannot produce (with high probability) any message M such that $H(M) = h$ in less than $\mathcal{O}(2^n)$ calls to $H(\cdot)$.

Definition 2 (Merkle-Damgård structure ($\mathcal{MDH}$)). *The Merkle-Damgård structure [5, 11] is a structure of an iterative hash function. Given a compression function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ that takes an n-bit chaining value and an m-bit message block and transforms them into a new n-bit chaining value, $\mathcal{MDH}^f$ is defined as follow: For an input message M:*

1. Padding step[1]
   (a) Concatenate '1' at the end of the message.
   (b) Let b be the number of bits in the message, and ℓ be the number of bits used to encode the message length in bits.[2] Pad a sequence of $0 \leq k < m$ zeros, such that $b + 1 + k + \ell \equiv 0 \pmod{m}$.
   (c) Append the message with the original message length in bits, encoded in ℓ bits.
2. Divide the message to blocks of m bits, so if the length of padded message is $L \cdot m$ then
$$M = m_0||m_1||\ldots||m_{L-1}.$$
3. The iterative process starts with a constant IV, denoted by h_{-1}, and it updated in every iteration, according to the appropriate message block m_i (for $0 \leq i \leq L-1$), to new chaining value:
$$h_i = f(h_{i-1}, m_i).$$
4. The output of this process is:
$$\mathcal{MDH}^f(M) = h_{L-1}$$

The process is depicted in Fig. 1.

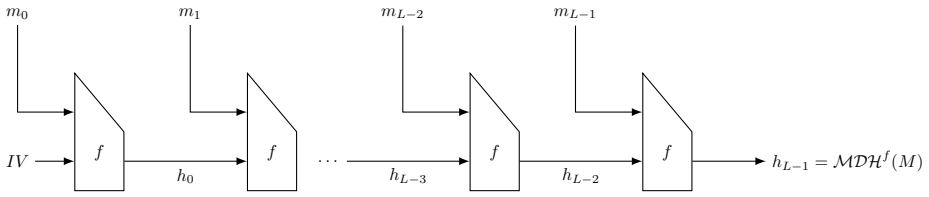

Fig. 1. The Merkle-Damgård Structure

Merkle [11] and Damgård [5] proved that if the compression function is collision-resistant then the whole structure (when the padded message includes the original message length) is also collision-resistant. Although the Merkle-Damgård structure is believed to be secure also from second pre-image attacks, in practice it is not [1, 2, 6, 9].

[1] We describe here the standard padding step done in many real hash functions such as MD5 and SHA1. Other variants of this step exist, all aiming to achieve prefix-freeness.

[2] It is common to set $2^\ell - 1$ as the maximal length of a message.

Definition 3. *Let $G = (V, E)$ be a directed graph. A directed edge from v to u is denoted by (v, u). For each $v \in V$ we define the in-degree of v, denoted by $d_{in}(v)$, to be the number of edges that ingoing to v, and the out-degree of v, denoted by $d_{out}(v)$, to be the number of edges that outgoing from v.*

Definition 4 (Galton-Watson Branching Process). *A Galton-Watson branching process is a stochastic process that illustrates a population increasing, which starts from one individual in the first state S_0, and for each $t \in \mathbb{N} \cup \{0\}$ each individual from S_t produces $i \in \mathbb{N} \cup \{0\}$ offsprings for the next state S_{t+1} according to a fixed probability distribution. Formally, a Galton-Watson branching process is defined as a Markov chain $\{Z_t\}_{t \in \mathbb{N} \cup \{0\}}$:*

1. *Let $Z_0 := 1$.*
2. *For each $(i, t) \in \mathbb{N} \times \mathbb{N}$ let X_i^t be a random variable follows a fixed probability distribution $P : \mathbb{N} \cup \{0\} \to [0, 1]$ with expected value $\mu < \infty$.*
3. *Define inductively:*

$$\forall t \in \mathbb{N} : Z_t := \sum_{1 \le i \le Z_{t-1}} X_i^t.$$

The random variable X_i^t represents the number of offspring produced by the i'th element (if there is one) of the Z_{t-1} elements from the time $t - 1$.

A central issue in the theory of branching processes is ultimate extinction, i.e, the event of some $Z_t = 0$. One can see that $E(Z_t) = \mu^t$. Still, even for $\mu \ge 1$ as long as $Pr[X_i = 0] > 0$ the ultimate extinction is an event with a positive probability. To study such events of ultimate extinction we need to study their probability, given by

$$\lim_{t \to \infty} Pr[Z_t = 0] = Pr[\exists t \in \mathbb{N} : Z_t = 0].$$

In [3] Athreya and Ney show that the probability of ultimate extinction is the smallest fixed point $x \in [0, 1]$ of the P's moment-generating function $f_P(x)$. For example, if $X_i^t \sim Poi(\lambda)$, then the probability of ultimate extinction is the smallest solution $x \in [0, 1]$ of $e^{\lambda(x-1)} = x$.

3 The Kite Generator

In [1] Andreeva et al. suggest a method to generate second pre-images for long messages of 2^k blocks. Using an expensive precomputation of 2^{n+1} compression function calls, the online complexity of their attack is $\max(\mathcal{O}(2^k), \mathcal{O}(2^{\frac{n-k}{2}}))$ time and $\mathcal{O}(2^{n-k})$ memory.

3.1 The Attack's steps

The Precomputation. In the precomputation the adversary constructs a data structure called the *kite generator*, which is a strongly connected directed graph with 2^{n-k} vertices. The vertices are labeled by chaining values and the directed

edges by message blocks which lead one chaining value to another. Given two chaining values a_1, a_2 covered by the kite generator, this structure allows to create a message of almost any desired length that connects a_1 to a_2.

To construct the kite generator, the adversary picks a set A of 2^{n-k} different chaining values, containing the IV. For each chaining value $a \in A$ he finds two message blocks $m_{a,1}, m_{a,2}$ such that $f(a, m_{a,1}), f(a, m_{a,2}) \in A$. We note that for each chaining value $a \in A$, $d_{out}(a) = 2$, and therefore $\forall a \in A : E[d_{in}(a)] = 2$.

The Online Phase. In the online phase, given a long message M, the adversary computes $H(M)$ and finds, with high probability, a chaining value h_i, for $n-k < i < 2^k$, such that $h_i \in A$. Now the adversary creates, using the kite generator, a sequence of i message blocks, starting from the IV, that leads to h_i. It is done in the following steps:

1. The adversary performs a random walk in the kite generator of $i-(n-k)$ message blocks, from the IV. To do so, the adversary starts from the IV and chooses an arbitrary message block $m \in \{m_{IV,1}, m_{IV,2}\}$ and traverse to the next chaining value $h_1 = f(IV, m)$. The adversary continues in the same manner $i-(n-k)-1$ times, until $h_{i-(n-k)}$ is reached. Denote the concatenation of the chosen message blocks by s_1.
2. The adversary computes all the $2^{\frac{n-k}{2}}$ chaining values reachable from $h_{i-(n-k)}$ by walking $\frac{n-k}{2}$ steps in the kite generator.
3. The adversary computes all the expected $2^{\frac{n-k}{2}}$ chaining values that may lead to h_i by walking back in the kite generator $\frac{n-k}{2}$ steps from h_i.
4. The adversary looks for a collision between these two lists (due to the birthday paradox, such a collision is expected with high probability). Denote the concatenation of the message blocks yielding from $h_{i-(n-k)}$ to the common chaining value by s_2, and the concatenation of the message blocks yielding from the common chaining value to h_i by s_3.

The concatenation $s_1||s_2||s_3$ is a sequence of i message blocks that leads from the IV to h_i, as desired. Now, the adversary creates a second pre-image:

$$M' = s_1||s_2||s_3||m_{i+1}||\cdots||m_k.$$

Fig. 2 illustrates the attack.

3.2 The Attack Complexity

The Precomputation Complexity. As described in Section 3.1, to construct the kite generator, the adversary has to find, for each chaining value $a \in A$, two message blocks $m_{a,1}, m_{a,2}$ such that $f(a, m_{a,1}), f(a, m_{a,2}) \in A$. To do so, he generates $2 \cdot 2^k$ message blocks, each leads to one of the 2^{n-k} chaining values of A with probability of 2^{-k}. Hence, is expected to find two such message blocks, and the total complexity is about $2 \cdot 2^k \cdot 2^{n-k} = 2^{n+1}$ compression function calls.[3]

[3] Note that using this method $d_{out}(a)$ follows a $Poi(2)$ distribution, and about 13% of the chaining values are expected to have $d_{out}(a) = 0$. To solve this issue, it is

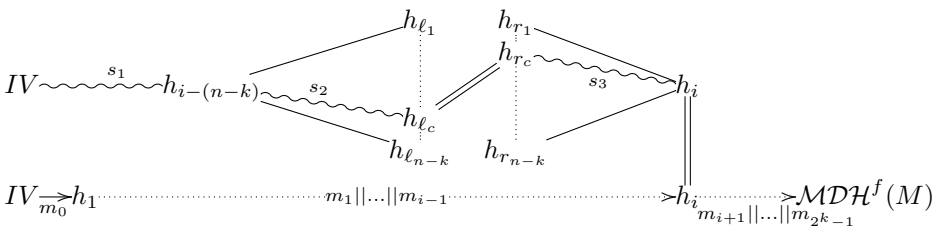

Fig. 2. A Second Pre-Image Using a Kite Generator.

The online Complexity. First of all, the memory used to store the kite generator is $\mathcal{O}(2^{n-k})$. Second, the online phase has two steps:

1. The adversary should compute the M's chaining values to find the common chaining value with the kite generator's chaining values. This step requires $\mathcal{O}(2^k)$ compression function calls.
2. The adversary should find a collision between the two lists described in Section 3.1. Since each list contains about $2^{\frac{n-k}{2}}$ chaining values, this step requires $\mathcal{O}(2^{\frac{n-k}{2}})$ time and memory.[4]

Thus, the online time complexity is

$$\max(\mathcal{O}(2^k), \mathcal{O}(2^{\frac{n-k}{2}})),$$

and the online memory complexity is

$$\mathcal{O}(2^{n-k}).$$

4 A Problem in the Construction of the Kite Generator

4.1 On the Inaccuracy of Andreeva et al.'s Analysis

As described in Section 3.1, for constructing the kite generator Andreeva et al. [1] suggest to find from each chaining value of A two message blocks, each of them

possible to generate for each chaining value as many message blocks as needed to find two out-edges. Now, the average time complexity needed for a chaining value a is 2^{k+1}. The actual running time for a given chaining value is the sum of two geometric random variables with mean 2^k each. Hence, the total running time is the sum of 2^{n-k+1} geometric random variables $X_i \sim Geo(2^{-k})$. Since $\sum_{i=1}^{2^{n-k+1}}(X_i-1) \sim NB(2^{n-k+1}, 1-2^{-k})$, then $\sum_{i=1}^{2^{n-k+1}} X_i \sim 2^{n-k+1}+NB(2^{n-k+1}, 1-2^{-k})$. Therefore, $E[\sum_{i=1}^{2^{n-k+1}} X_i] = 2^{n-k+1} + \frac{(1-2^{-k})2^{n-k+1}}{2^{-k}} = 2^{n+1}$ with a standard deviation of $\frac{\sqrt{2^{n-k+1}(1-2^{-k})}}{2^{-k}} \leq 2^{\frac{n+k+1}{2}}$.

[4] Andreeva et al. [1] note that it is possible to find the common chaining value by a more sophisticated algorithm which requires the same time but negligible additional memory, using memoryless collision finding. Our findings affect these variants as well.

leads to a chaining value of A. They claim that as $\forall a \in A : E[d_{in}(a)] = 2$, the resulting graph is strongly connected.

We agree with their claim that due to the fact that $d_{out}(a) = 2$ then $\forall a \in A : E[d_{in}(a)] = 2$, but we claim that their conclusion, that the resulting graph is strongly connected, is wrong. The actual distribution of $d_{in}(a)$ can be approximated by $d_{in}(a) \sim Poi(2)$, as the number of entering edges follows a Poisson distribution with a mean value of 2. Hence, for each chaining value $a \in A$:

$$Pr[d_{in}(a) = 0] = \frac{e^{-2} \cdot 2^0}{0!} = e^{-2}.$$

Thus, about $2^{n-k} \cdot e^{-2} \approx 13\%$ of the chaining values in the kite generator are expected to have $d_{in}(a) = 0$. Obviously, the resulting graph is not strongly connected.

Moreover, there are more h_i's in the kite generator for which the attack fails. The construction of the "backwards" tree from h_i is a branching process with $Poi(2)$ offspring (see Def. 4). Therefore, an ultimate extinction of the branching process suggests that h_i cannot be connected to, and the online phase fails. According to the branching process theorems [3], the probability of ultimate extinction in a branching process with offspring distribute according a distribution P is the smallest fixed point $x \in [0, 1]$ of the moment-generating function of P. In our case the distribution is $Poi(2)$, and the moment-generating function is $f(x) = e^{2(x-1)}$. Hence, the probability of ultimate extinction is the smallest solution $x \in [0, 1]$ of $e^{2(x-1)} = x$. Using numerical computation we get that $x \approx 0.2032$. It means that in about 20% of the cases the "backwards" tree is limited. We note that usually this extinction happens very quickly. For example, about 85% of the "extinct" h_i do so in one or zero steps (i.e., their backwards tree is of depth of at most 1).

To fix this problem we need that $E[d_{in}(a)] = n - k$, and then the expected number of chaining values $a \in A$ with $d_{in}(a) = 0$ is $2^{n-k} \cdot e^{-(n-k)} \ll 1$. The naive approach to do so, is to generate from each chaining value $a \in A$, $n - k$ message blocks, $m_{a,1}, m_{a,2}, \ldots, m_{a,n-k}$, for which $f(a, m_{a,i}) \in A \setminus \{a\}$. Using this approach, the complexity of the precomputation increases to $(n - k) \cdot 2^n$ compression function calls.

A different approach for fixing the problem is to increase the kite generator by adding vertices such that the intersection between a message of length 2^k and the kite generator is sufficiently large (i.e., that there are several joint chaining values). Hence, even if some of the pairs of the joint chaining values fail to connect through the kite generator, there is a sufficient number of pairs that do connect. This approach does not increase the precomputation time beyond 2^{n+1} (as the additional vertices in the kite generator reduce the "cost" of connecting any vertex). At the same time, it increases the memory complexity of the attack. We do not provide a full analysis of this approach given the improved attack of Section 5 which does not require additional memory, and enjoys a smaller time complexity.

5 A New Method for Constructing Kite Generators

In the construction described in Section 3.1, the set A of the chaining values is chosen arbitrarily. We now suggest a new method for choosing the chaining values in order to optimize the complexity of constructing a kite generator. Our main idea is to define the set A iteratively in a manner that ensures that $d_{in}(a) \geq 1$ (for all but one chaining value, the IV).

Construction. For the reading convenience we consider two different message blocks m_1, m_2.[5] The following steps are required:

1. Let $L_0 := \{h_0 = IV\}$.
2. Set the second layer $L_1 = \{h_1 = f(IV, m_1), h_2 = f(IV, m_2)\}$.
3. Continue by the same method to set
$$L_i = \{f(h, m) \mid h \in L_{i-1}, m \in \{m_1, m_2\}\}, \forall 1 \leq i \leq n-k-1$$
until L_{n-k-1} is generated.
4. Set
$$A = \bigcup_{i=0}^{n-k-1} L_i.$$
Note that[6] $|A| = \sum_{i=0}^{n-k-1} 2^i = 2^{n-k} - 1$.
5. Finally, for each chaining value a reached in the last layer L_{n-k-1}, look for two message blocks $m_{a,1}, m_{a,2}$ (probably different than m_1, m_2) that lead to some chaining value $b \in A$.

Fig. 3 illustrates the construction of A.

The advantage of this method is that for each chaining value $IV \neq a \in A$, there exists another chaining value $b \in A$ and a message block m_b such that

$$f(b, m_b) = a$$

i.e.,

$$d_{in}(a) \geq 1.$$

The case of $d_{in}(IV) = 0$ is not problematic, since we need the IV in the kite generator only as the source of the random walking, and it is done only with the out-edges. In addition, In this method of constructing A, each chaining value $a \neq IV$ follows $d_{in}(a) \sim 1 + Poi(1)$. It implies that $\forall a \neq IV : Pr[d_{in}(a) = 0] = 0$, and therefore the probability of ultimate extinction in the branching process defined by the backwards tree is 0.

[5] It is not necessary to use only two different message blocks in the setting, but it is possible since they are used for different chaining values.

[6] With high probability we expect some collisions in A. This can be easily solved during the construction: If a chaining value $f(h_i, m_j)$ is already generated, replace the message block m_j one by one until a new chaining value is reached. It is easy to see that the additional time complexity is negligible.

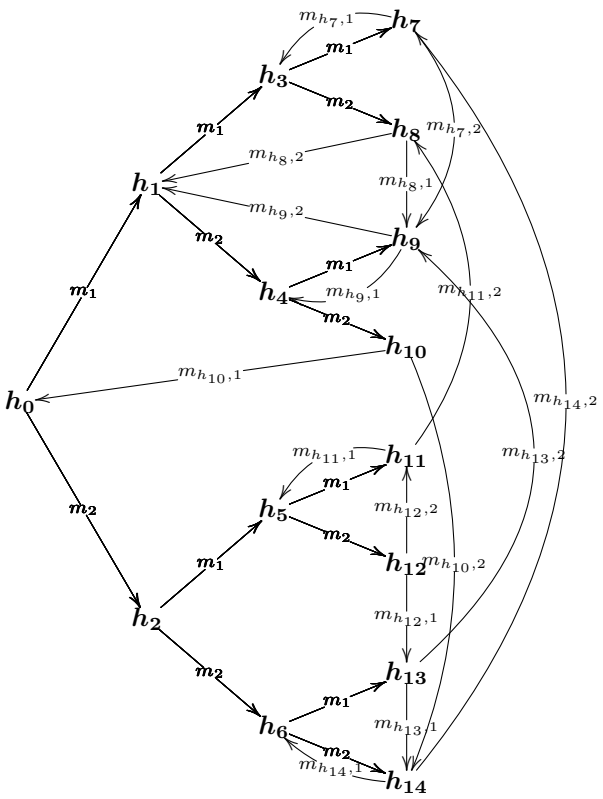

Fig. 3. An Example for an Iterative Construction of A

Analysis. Steps 1-4 generate arbitrary message blocks for each reached chaining value until about 2^{n-k} chaining values are reached. They require about 2^{n-k} compression function calls. Step 5, of finding two out-edges from each chaining value that reached in the last layer L_{n-k-1}, requires about $2 \cdot 2^{n-k-1} \cdot 2^k = 2^n$ compression function calls.[7] Thus, the precomputation complexity is about

$$2^n + 2^{n-k} \approx 2^n$$

[7] Again, in this step we actually need to generate for each chaining value as many message blocks as needed to find two out-edges. Now, the average time complexity needed for a chaining value a is 2^{k+1}. The actual running time for a given chaining value is the sum of two geometric random variables with mean 2^k each. Hence, the total running time is the sum of 2^{n-k} geometric random variables $X_i \sim Geo(2^{-k})$. Since $\sum_{i=1}^{2^{n-k}}(X_i - 1) \sim NB(2^{n-k}, 1-2^{-k})$, then $\sum_{i=1}^{2^{n-k}} X_i \sim 2^{n-k} + NB(2^{n-k}, 1-2^{-k})$. Therefore, $E[\sum_{i=1}^{2^{n-k}} X_i] = 2^{n-k} + \frac{(1-2^{-k})2^{n-k}}{2^{-k}} = 2^n$ with a standard deviation of $\frac{\sqrt{2^{n-k}(1-2^{-k})}}{2^{-k}} \leq 2^{\frac{n+k}{2}}$.

compression function calls. It means that our method not only ensures that the resulting graph is strongly connected, but is also more efficient than the original method.

Improvement I. As additional improvement, we can reduce the complexity of Step 5 described above by finding only one out-edge from each chaining value in the last layer L_{n-k-1}. Now, each chaining value $a \neq IV$ follows $d_{in}(a) \sim 1 + Poi(0.5)$, and the branching process does not extinct. To use this improvement, we need to slightly change the online phase: We need to increase the length of the sequences s_2, s_3 mentioned in Section 3.1 to about $\log_{\frac{3}{2}}(2) \cdot \frac{n-k}{2}$ (instead of $\frac{n-k}{2}$) to find, with high probability, a common chaining value. Using this improvement, the complexity of the precomputation is reduced by a factor of 2 to about 2^{n-1} compression function calls. There is no change in the online time complexity.

6 Adapting our New Method to Dithered Merkle-Damgård

6.1 Dithered Merkle-Damgård

The main idea of the dithered Merkle-Damgård structure [13] is to perturb the hashing process by using an additional input to the compression function. This additional input is formed by taking elements of a fixed dithering sequence. Using this additional input, the compression of a message block depends on its position in the whole message. Thus, it decreases the adversary's control on the input of the compression function. Using the dithered Merkle-Damgård structure, some attacks such as the Dean's attack [6] and the Kelsey-Schneier's expandable-messages attack [9] are mitigated.

In order to use the dithered sequence for any message with the maximal number of message blocks in the hash function, it is reasonable to consider an infinite sequence. Let $\mathcal{B}$ be a finite alphabet, and let z be an infinite sequence over $\mathcal{B}$ and let z_i be the i'th symbol of z. The dithered Merkle-Damgård construction is obtained by replacing the iterative chaining value defined in the original Merkle-Damgård structure (see Def. 2) with

$$h_i = f(h_{i-1}, m_i, z_i).$$

6.2 Adapting the Kite Generator to Dithering Sequence

As described in Section 3.1, the adversary could not know in advance the position of the message blocks to be used in the second pre-image. Thus, in order to allow the use of the kite generator at each position of the message, the adversary should consider any factor of z. To do so, Andreeva et al. [1] adapt the precomputation phase as follows: For each chaining value $a \in A$ and for each symbol $\alpha \in \mathcal{B}$ the adversary looks for two message blocks $m_{a,\alpha,1}, m_{a,\alpha,2}$ s.t.

$f(a, m_{a,\alpha,1}, \alpha), f(a, m_{a,\alpha,2}, \alpha) \in A$. Hence, The complexity of the precomputation using the original method is about

$$2 \cdot |\mathcal{B}| \cdot 2^n$$

compression function calls.

The same problem mentioned in Section 4.1 carries over to this case as well. As described in Section 4.1, in order to fix the inaccuracy that the resulting graph is not strongly connected, the adversary should generate about $n-k$ such message blocks for each chaining value and for each dithered symbol. Hence, the complexity of the precomputation is increased to about

$$(n-k) \cdot |\mathcal{B}| \cdot 2^n$$

compression function calls.

6.3 Adapting our Method

Construction. As described in Section 5, the main idea of our new method is to choose the chaining values of A by generating two message blocks for each reached chaining value, starting from the IV. In order to adapt it to the dithered Merkle-Damgård structure, the following steps are required:

1. Choose an arbitrary symbol $\alpha \in \mathcal{B}$, and construct the kite generator using this symbol only, according to our new method.
2. Use the original method to complete the kite generation for the remaining symbols of $\mathcal{B}$, i.e., for each chaining value $a \in A$ and for each symbol $\alpha \neq \beta \in \mathcal{B}$, look for $n-k$ message blocks that lead to another chaining value of A.

Analysis. The complexity of the first step, of constructing the kite generator using one symbol, is similar to the one in Section **??**, i.e., about 2^{n-1} compression function calls (using the improved method). The complexity of the second step, of completing the kite generator for the remaining symbols, is about $(n-k)\cdot(|\mathcal{B}|-1)\cdot 2^n$ compression function calls. Thus, the total complexity of the precomputation is about

$$((n-k)\cdot(|\mathcal{B}|-1)+0.5)\cdot 2^n$$

compression function calls.

6.4 Improvement II

In Section 6.3 we adapted our new method while considering the probability of ultimate extinction in the construction of the "backwards" tree tends to zero. We now show that by allowing a small probability of ultimate extinction, denoted by p, we can reduce the complexity as follows. We look for a $\lambda(p)$ such that

the probability of ultimate extinction in a branching process with $Poi(\lambda(p))$ offspring is p. According to the branching process theorems [3] we need that

$$e^{\lambda(p)(1-p)} = p$$

that implies

$$\lambda(p) = \frac{\ln(p)}{1-p}.$$

For examples, $\lambda(0.01) \approx 4.65$, and $\lambda(0.001) \approx 6.91$. It means that in the construction described in Section 6.3, we can replace the second step of looking for $n-k$ message blocks per symbol for each chaining value, by looking for such a constant number. Thus, consider $p = 0.001$, the complexity of the precomputation is reduced to about

$$(6.91 \cdot (|\mathcal{B}| - 1) + 0.5) \cdot 2^n$$

compression function calls.

7 Summary

As a concluding discussion, we note that when the kite generator has 2^{n-k} chaining values and the message is of length 2^k blocks, one should expect the kite generator to contain one of the message's chaining values with probability 63%, which translates to about 50% success rate. The way to fix this issue is trivial — increase the size of the kite generator. Multiplying the number of nodes in the kite generator by a factor of 2, reduces the probability of disjoint sets of chaining values from $1/e \approx 37\%$ to merely $1/e^2 \approx 13.5\%$, and this rate can be further reduced to as small probability as the adversary wishes.[8] However, when the kite generator is not strongly connected, as our analysis shows, the success probability of the original is upper bounded by 80%, no matter how large the kite generator is taken to be.

To conclude, in this work we pointed out an inaccuracy in the analysis of the construction of the *kite generator* suggested by Andreeva et al. [1]: The *kite generator* is not strongly connected, and thus the online phase fails in probability of at least 20%. We showed that to fix the inaccuracy, we need to increase the complexity of the construction phase by a factor of $\frac{n-k}{2}$. We then suggested a new method to optimize the construction of the *kite generator* that is both correct and more efficient than the original method. Finally, we adapted the fixing analysis and our new method to the dithered Merkle-Damgård construction.

For comparison, we present in Table 1 the number of required compression function calls to construct a *kite generator*, using the different methods, for the Merkle-Damgård structure and its dithered variant.

[8] This issue happens also in the online phase, when the adversary looks for common chaining values between the two lists described in Section 3.1. The fixing is similarly – increase the size of these lists accordingly.

Method	Complexity	
	Merkle-Damgård	Dithered Merkle-Damgård
Andreeva et al. [1] †	2^{n+1}	$\lvert\mathcal{B}\rvert \cdot 2^{n+1}$
Our fixed analysis	$(n-k) \cdot 2^n$	$(n-k) \cdot \lvert\mathcal{B}\rvert \cdot 2^n$
Our new method	2^n	$((n-k) \cdot (\lvert\mathcal{B}\rvert - 1) + 1) \cdot 2^n$
Improvement I	2^{n-1}	$((n-k) \cdot (\lvert\mathcal{B}\rvert - 1) + 0.5) \cdot 2^n$
Improvement II	Not relevant	$(6.91 \cdot (\lvert\mathcal{B}\rvert - 1) + 0.5) \cdot 2^n$

† — Andreeva et al.'s analysis is inaccurate.

Table 1. Comparing the Complexities of the Different Methods.

Acknowledgements

The research of Ariel Weizman was supported by the European Research Council under the ERC starting grant agreement n. 757731 (LightCrypt) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

1. Elena Andreeva, Charles Bouillaguet, Orr Dunkelman, Pierre-Alain Fouque, Jonathan Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. New second-preimage attacks on hash functions. *J. Cryptology*, 29(4):657–696, October 2016.
2. Elena Andreeva, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. Second preimage attacks on dithered hash functions. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2008.
3. Krishna B. Athreya and Peter E. Ney. Branching processes. Dover Books on Mathematics, chapter 1, pages 1–8. Dover Publications, 2004.
4. Simon R. Blackburn, Douglas R. Stinson, and Jalaj Upadhyay. On the complexity of the herding attack and some related attacks on hash functions. *Designs, Codes and Cryptography*, 64(1-2):171–193, 2012.
5. Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
6. Richard Drews Dean. *Formal aspects of mobile code security*. PhD thesis, Princeton University, Princeton, 1999.
7. Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004.

8. John Kelsey and Tadayoshi Kohno. Herding hash functions and the Nostradamus attack. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2006.
9. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2n work. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005.
10. Tuomas Kortelainen and Juha Kortelainen. On diamond structures and Trojan message attacks. In *Advances in Cryptology - ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 524–539. Springer, 2013.
11. Ralph C. Merkle. One way hash functions and DES. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
12. National Institute of Standards and Technology. Secure hash standard. *FIPS, PUB*, 17:3–180, 1995.
13. Ronald L Rivest. Abelian square-free dithering for iterated hash functions. In *ECrypt Hash Function Workshop, June*, volume 21, 2005.
14. Ariel Weizmann, Orr Dunkelman, and Simi Haber. Efficient construction of diamond structures. In *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017*, volume 10698 of *Lecture Notes in Computer Science*, pages 166–185, 2017.