Cryptanalysis of MDC-2[⋆]

Lars R. Knudsen[1], Florian Mendel[2], Christian Rechberger[2], and Søren S. Thomsen[1]

[1] Department of Mathematics, Technical University of Denmark
Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark
[2] Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

Abstract. We provide a collision attack and preimage attacks on the MDC-2 construction, which is a method (dating back to 1988) of turning an n-bit block cipher into a $2n$-bit hash function. The collision attack is the first below the birthday bound to be described for MDC-2 and, with $n = 128$, it has complexity $2^{124.5}$, which is to be compared to the birthday attack having complexity 2^{128}. The preimage attacks constitute new time/memory trade-offs; the most efficient attack requires time and space about 2^n, which is to be compared to the previous best known preimage attack of Lai and Massey (Eurocrypt '92), having time complexity $2^{3n/2}$ and space complexity $2^{n/2}$, and to a brute force preimage attack having complexity 2^{2n}.
Keywords: MDC-2, hash function, collision, preimage

1 Introduction

MDC-2 is a method of constructing hash functions from block ciphers, where the output size of the hash function is twice the size of the block cipher (hence it is called a *double-length construction*). MDC-2 was developed at IBM in the late 80s. A conference paper by IBM researchers Meyer and Schilling from 1988 describes the construction [21]. A patent was filed in August 1987, and the patent was issued in March 1990 [1]. The construction was standardised in ISO/IEC 10118-2 in 1994 [9]. It is mentioned in great detail in both the Handbook of Applied Cryptography [20, Alg. 9.46] and in the Encyclopedia of Cryptography and Security [27, pp. 379–380]. Furthermore, it is in practical use (see e.g., [10, 15, 26]).

Since publication, there seems to have been a wide belief in the cryptographic community that given an ideal block cipher, MDC-2 provides a collision resistant hash function. By this we mean that given an n-bit block cipher (thus yielding a $2n$-bit hash function), the required effort to find a collision in the hash function is expected to be 2^n. However, there is no proof of this property. The only proof that collision resistance is better than $2^{n/2}$, as offered by many simpler (single-length) constructions, is due to Steinberger [25], who showed that for MDC-2 based on an ideal cipher, an adversary asking less than $2^{3n/5}$ queries has only a negligible chance of finding a collision.

In this paper we provide the first collision attack on MDC-2 which breaks the birthday bound. The attack makes no non-standard assumptions on the underlying block cipher. When applied to an instantiation of MDC-2 with e.g., a 128-bit block cipher (see e.g., [28]), the attack has complexity about $2^{124.5}$, which is better than the expected 2^{128} collision resistance for an ideal 256-bit hash function.

We also present improved preimage attacks on MDC-2. The previous best known preimage attack, first described by Lai and Massey [16], has time complexity about $2^{3n/2}$ and requires around $2^{n/2}$ memory. In this paper we provide a range of time/memory trade-offs, the fastest of which is significantly faster than the Lai/Massey attack. We describe attacks of any time complexity from 2^n to 2^{2n}. The memory requirements are such that the product of the time

[⋆] This work has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

and space complexities is always around 2^{2n}. Hence, our most efficient preimage attack has time and space complexity about 2^n.

Finally, we describe how to use the preimage attack to find multicollisions faster than by the previous best known multicollision attack of Joux [11].

Related work. As mentioned, Lai and Massey described [16] a preimage attack on MDC-2 of complexity around $2^{3n/2}$. Knudsen and Preneel gave [14] a preimage attack on MDC-4 (a stronger and less efficient variant of MDC-2, to which the attacks described in this paper do not apply) of complexity $2^{7n/4}$. Steinberger proved [25] a lower bound of $2^{3n/5}$ for collision resistance of MDC-2 in the ideal cipher model.

Our attacks in fact apply to a larger class of hash function constructions based on block ciphers (see Section 2.1). Knudsen, Lai and Preneel described [13] collision and preimage attacks on all block cipher based hash function constructions of rate 1, meaning that one message block is processed per block cipher call. These attacks do not apply to MDC-2 (having rate 1/2).

Recently, a number of new double-length constructions have been proposed. At FSE 2005, Nandi et al. [23] proposed a rate 2/3 scheme, and they proved that finding a collision requires at least $2^{2n/3}$ queries. Later the same year (Indocrypt 2005), Nandi [22] introduced a class of rate 1/2 double-length schemes, all instances of which having optimal collision resistance 2^n. At Asiacrypt 2005, Lucks [18] proposed the double-pipe scheme as a failure-friendly design, meaning that collision resistance is retained even if the underlying compression function slightly fails to be collision resistant. The scheme maintains two chains, which are combined at the end, and hence is in fact a single-length scheme. However, by omitting the merging at the end one has a double-length scheme, which is optimally collision resistant. Hirose [8] proposed (FSE 2006) a collision resistant double-length scheme, based on an n-bit block cipher accepting keys of more than n bits. The rate depends on the key size. For all these schemes, the security proof is based on the assumption that the underlying primitive (compression function or block cipher) is secure. Our attacks do not apply to any of the schemes mentioned here.

Hellman has described a generic method to find a preimage of a $2n$-bit hash function with runtime $2^{4n/3}$ [7]. The caveat is that (apart from requiring $2^{4n/3}$ memory) a precomputation of cost 2^{2n} is needed. The preimage attacks on MDC-2 that are described in this paper are on a much better time/memory trade-off curve, and do not require a 2^{2n} precomputation.

2 Preliminaries

The collision attack presented in this paper makes use of *multicollisions.*

Definition 1. *Let f be some function. An r-collision for f is an r-set $\{x_1, \ldots, x_r\}$ such that $f(x_1) = \ldots = f(x_r)$. A multicollision is an r-collision for some $r > 1$. A 2-collision is known simply as a collision.*

Consider the classical occupancy problem (see e.g., [5]) consisting of randomly throwing q_1 balls into 2^n urns, where it is assumed that each of the 2^{nq_1} possible outcomes is equally likely. In order for the probability that at least one urn contains at least r balls to be $1 - 1/e$, one must throw about

$$q_1 = (r!2^{n(r-1)})^{1/r} \tag{1}$$

balls in total [5, IV,(2.12)]. The classical occupancy problem can be translated into the problem of finding an r-collision for a sufficiently random n-bit function f. Hence, this task has expected complexity q_1 as given by (1). In the following we shall use this expression as an estimate for the complexity of finding an r-collision.

We note that a standard birthday collision attack has complexity $2^{(n+1)/2}$, according to (1) with $r = 2$. With $2^{n/2}$ queries a collision is found with probability about $1 - e^{-1/2} \approx 0.39$.

2.1 Description of the MDC-2 construction

MDC-2 was originally defined using DES [24] as the underlying block cipher. Here, we think of MDC-2 as a general double-length construction method for hash functions based on block ciphers. For ease of presentation we shall assume that keys and message blocks are of the same size, even if this is in fact not the case for DES. In Appendix A, we discuss this special case.

Let $E_K(m)$ denote the encryption under some block cipher (assumed to be secure) of plaintext m using the key K. If X is an n-bit string, then we let X^L denote the leftmost $n/2$ bits of X, and we let X^R denote the rightmost $n/2$ bits of X. Given E, MDC-2 defines a $2n$-bit hash function (with some given, distinct initial values H_0 and $\tilde{H}_0$) as follows. Split the message M (assumed to be appropriately padded) into t blocks $m_1, \ldots, m_t$, and do, for each i from 1 to t, the following ('$\|$' denotes concatenation).

$$\begin{aligned} V &= E_{H_{i-1}}(m_i) \oplus m_i \\ \tilde{V} &= E_{\tilde{H}_{i-1}}(m_i) \oplus m_i, \end{aligned}$$

followed by

$$\begin{aligned} H_i &= V^L \| \tilde{V}^R \\ \tilde{H}_i &= \tilde{V}^L \| V^R. \end{aligned}$$

The output is $H_t \| \tilde{H}_t$. See also Figure 1. In other words, the chaining variables H_{i-1} and $\tilde{H}_{i-1}$

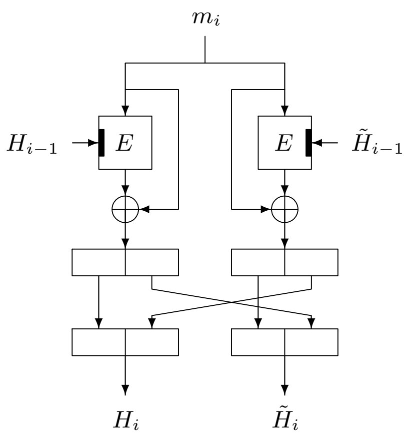

Fig. 1: The MDC-2 construction.

are used as keys in two block cipher calls, which each encrypt the message block m_i, and subsequently xor the resulting ciphertexts with m_i. The two right halves of the results are then swapped to obtain the next pair of chaining variables. In what follows, these steps will be called *an iteration.*

In the original description of MDC-2 [21], two bits of each of the two keys H_{i-1} and $\tilde{H}_{i-1}$ were fixed. This had two implications. First of all, all known weak and semi-weak keys of DES were ruled out, and secondly, this measure ensured that the two keys were always different. There seems to be no strong consensus that fixing key bits is a necessary security measure when MDC-2 is based on some other block cipher for which weak keys are not believed to exist. However, one might argue that ensuring that the two keys are different increases security – although this practice also has a cost in terms of security: the amount of state passed on from one iteration to the next is less than $2n$ bits. The attacks presented in this paper can be applied regardless of whether or not some key bits are fixed. However, the discussion of Section 6 assumes that no key bits are fixed.

A generalisation. We may generalise the MDC-2 construction. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be any function, and let g be any (efficiently invertible) bijection from $2n$ bits to $2n$ bits. Then a generalised construction is the following.

$$\begin{aligned} W &= f(H_{i-1}, m_i) \| f(\tilde{H}_{i-1}, m_i) \\ H_i \| \tilde{H}_i &= g(W). \end{aligned} \tag{2}$$

See Figure 2. In standard terms, (2) defines a *compression function* $h : \{0,1\}^{3n} \to \{0,1\}^{2n}$. The

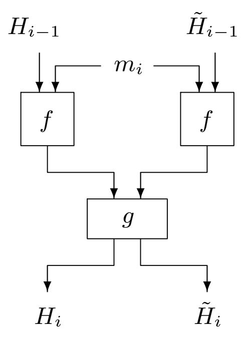

Fig. 2: The generalised MDC-2 construction.

attacks presented in this paper apply to any instance of this construction. Notice that MDC-2 has $f(x,y) = E_x(y) \oplus y$ and $g(a\|b\|c\|d) = a\|d\|c\|b$. In the following we shall use the notation of the generalised construction. We assume that evaluating g (both forwards and backwards) costs much less than evaluating f. Our complexity estimates will be in terms of compression function evaluations. For example, if an attack requires T calls of f, we shall count this as having time complexity $T/2$, since f is evaluated twice in the compression function.

3 The collision attack

The collision attack applies to any construction of the type (2). We use the notation of Section 2 in the following description of the collision attack.

1. Given initial chaining values H_0 and $\tilde{H}_0$, find an r-collision in H_1. Let the messages producing the r-collision be $m_1^1, \ldots, m_1^r$, and let the r ("random") values of $\tilde{H}_1$ be $\tilde{H}_1^1, \ldots, \tilde{H}_1^r$.
2. Let $\ell = 1$.
3. Choose the message block m_2^ℓ arbitrarily, and evaluate $W_j^\ell = f(\tilde{H}_1^j, m_2^\ell)$ for every j, $1 \le j \le r$. If $W_i^\ell = W_j^\ell$ for some $i \neq j$, $1 \le i,j \le r$, then a collision $(m_1^i \| m_2^\ell, m_1^j \| m_2^\ell)$ has been found. If not, increment ℓ and repeat this step.

See Figure 3. Step 1 requires finding an r-collision in an n-bit function. This is expected to take

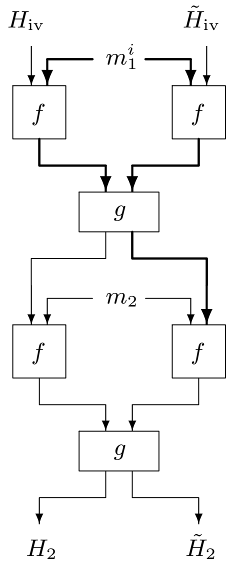

Fig. 3: The collision attack. Thick lines mean that there are r different values of this variable. Thin lines mean that there is only one.

time $q_1 = (r!2^{n(r-1)})^{1/r}$ as mentioned in Section 2. The probability of success in Step 3 is about $\binom{r}{2}2^{-n}$, since there are $\binom{r}{2}$ pairs of n-bit values, which may be equal. Hence, we expect to need to repeat Step 3 $2^n/\binom{r}{2}$ times. In each iteration we evaluate the encryption function r times. In the construction (2), f is evaluated twice per message block, and hence the r evaluations of f are equivalent to $r/2$ compression function evaluations. The total work required in Step 3 is therefore expected to be

$$q_2 = (r/2) \cdot 2^n / \binom{r}{2} = 2^n/(r-1).$$

The total work required is $q_1 + q_2 = (r!2^{n(r-1)})^{1/r} + 2^n/(r-1)$. Hence, we may choose r as the integer ≥ 2 that minimises this expression. Notice that q_1 is an increasing function of r, and q_2 is decreasing. By setting $q_1 = q_2$ one gets, very roughly, a time complexity around $(\log_2(n)/n)2^n$. However, it turns out that the best choice of r is not exactly the one where $q_1 = q_2$, as one might expect. Table 1 shows the best choices of r and the corresponding complexities for different sizes n of the block cipher.

Table 1: Time complexity of the collision attack on MDC-2 with an n-bit block cipher, compared to birthday complexity. For details in the case of MDC-2 based on DES ($n = 54$), see Appendix A.1.

n	r	Collision attack complexity	
		Section 3	Birthday
54	8	$2^{51.5}$	2^{54}
64	9	$2^{61.3}$	2^{64}
128	14	$2^{124.5}$	2^{128}
256	24	$2^{251.7}$	2^{256}

The probability of success of our attack with these complexities is about $1 - 1/e$ for Step 1, and the same probability for Step 3 when repeated $2^n/\binom{r}{2}$ times, in total $(1 - 1/e)^2 \approx 0.40$.

As mentioned in Section 2, the probability of success for the birthday attack with 2^n queries is about $1 - e^{-1/2} \approx 0.39$. Hence, we consider the comparisons fair.

4 Preimage attacks

A brute force preimage attack on MDC-2 (or on (2) in general) has time complexity $O(2^{2n})$ and space complexity $O(1)$. The previous best known preimage attack is due to Lai and Massey [16], and has time complexity $O(2^{3n/2})$ and space complexity $O(2^{n/2})$. Hence, for both attacks the product of the time complexity and the space complexity is $O(2^{2n})$. In the following subsection we describe a range of preimage attack time/memory trade-offs, for which the product of the time and the space complexities is at most $n2^{2n}$, but where time complexity can be anything between $O(n2^n)$ and $O(2^{2n})$. In Section 4.2 we describe how to reach a time and space complexity of $O(2^n)$.

4.1 An attack allowing for time/memory trade-offs

The attack uses pseudo-preimages, which are preimages of the compression function where both the chaining value and the message block can be chosen freely by the attacker. The attack can be outlined as follows.

1. Build a binary tree of pseudo-preimages with the target image $H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$ as root: the nodes are labelled with intermediate hash values, and each edge is labelled with a message block value meaning that this message block maps from the intermediate hash value at the child node to the intermediate hash value at the parent. The tree has (on average) two children for each node, and it has depth d meaning there are 2^d leaves.
2. From the initial value $H_{\mathrm{iv}} \| \tilde{H}_{\mathrm{iv}}$ of the hash function, find a message block that produces an intermediate hash value equal to one of the leaves in the tree from Step 1.

See Figure 4. The above technique clearly leads to a preimage consisting of a message block

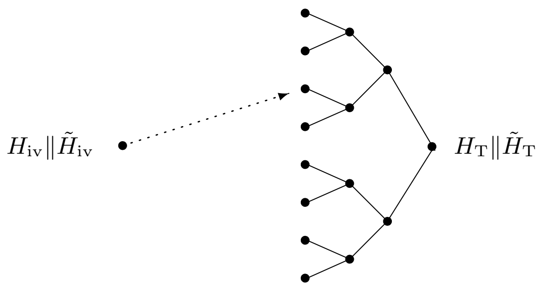
$H_{\mathrm{iv}} \| \tilde{H}_{\mathrm{iv}}$
$H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$

Fig. 4: A binary tree of pseudo-preimages of depth $d = 3$.

that maps to a leaf ℓ in the tree, and a sequence of d message blocks corresponding to the path in the tree that leads from the leaf ℓ to the root. Hence the total length of the message is $d + 1$ blocks.

The value of d determines the time/memory trade-off. We shall discuss concrete values of d later. The cost of Step 1 will be evaluated in the following. Since the tree has 2^d leaves, Step 2 is expected to take time 2^{2n-d}. In effect, by constructing the tree we produce 2^d new target images, which improves the efficiency of the final brute force search by a factor of 2^d. The memory requirements are $2^d + 2^{d-1} + \ldots + 1 = 2^{d+1} - 1$ intermediate hash values.

We note that the last message block, the one that maps to the target image, must contain proper padding for a message of $d+1$ blocks. If there are not enough degrees of freedom in the last block to both ensure proper padding and to find two pseudo-preimages, then a few initial steps (consisting of finding a small number of pseudo-preimages) are needed to ensure proper padding. It will become clear in the following that this only has a small effect on the total time complexity.

Constructing the tree (Step 1 above) is very time consuming for an ideal hash function. However, for the MDC-2 construction, there is an efficient method based on the following theorem.

Theorem 1. *Given a target hash value $H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$, a pseudo-preimage can be found in time at most 2^{n-1} with probability about $(1-1/e)^2$. By a pseudo-preimage we mean a pair $(H_{\mathrm{p}}, \tilde{H}_{\mathrm{p}})$ and a message block m such that $g(f(H_{\mathrm{p}}, m)\|f(\tilde{H}_{\mathrm{p}}, m)) = H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$.*

Proof. The method is the following. Let $U\|\tilde{U} = g^{-1}(H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}})$. Choose m arbitrarily, define $f_m(x) = f(x, m)$, and evaluate f_m on all $x \in \{0,1\}^n$. Referring again to the classical occupancy problem, when randomly throwing 2^n balls into 2^n urns, the probability that a given urn contains at least one ball is about $1-1/e$. Assuming that f_m is sufficiently random, this means that the probability that a given image has at least one preimage is about $1-1/e$, and additionally assuming independence, it means that the probability of finding at least one preimage of both U and $\tilde{U}$ is $(1-1/e)^2$. Let these preimages be H_{p} and $\tilde{H}_{\mathrm{p}}$, respectively. Then $g(f_m(H_{\mathrm{p}})\|f_m(\tilde{H}_{\mathrm{p}})) = H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$. Finally, the complexity of evaluating f_m 2^n times corresponds to 2^{n-1} compression function evaluations. $\square$

We note that for an ideal $2n$-bit compression function, the above task has complexity about 2^{2n}. The story does not finish with Theorem 1, however. Clearly, by evaluating a random n-bit function 2^n times, one finds on average one preimage for all elements of $\{0,1\}^n$. Thus, we obtain the following corollary.

Corollary 1. *Given t target hash values, in time 2^{n-1} one pseudo-preimage (on average) can be found for each target hash value. Here, t can be any number between 1 and 2^n.*

Proof. The technique is the same as above (we note that inverting g, which must be done 2^t times, is assumed to be a much simpler task than evaluating f). Since f_m is evaluated on all 2^n possible inputs, on average one preimage is found for each element of $\{0,1\}^n$. Therefore, again assuming independence, we also expect one preimage on average of each of the t target hash values. With respect to the complexity, we repeat that 2^n calls to f_m is equivalent to about 2^{n-1} compression function calls. $\square$

In the case of MDC-2, where g has a special form that allows to compute n bits of the output given only n bits of the input (and vice versa), t above can actually be 2^{2n} without affecting the complexity. The reason is that g (in this case) never has to be inverted more than 2^n times.

Due to Theorem 1 and Corollary 1, the tree described above can be efficiently constructed as follows (note that the tree will, in fact, not be binary, due to some nodes having no children, and others having more than two, but on average the number of children per node will be two):

Assign the value $H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$ of the target image to the root of the tree. Then find (in expected time 2^n) two pseudo-preimages of the target image by the method of Theorem 1 (applied twice with different message blocks m). This means the tree now contains the root and two children of the root. Then find two pseudo-preimages of each of the two children of the root. This also takes time 2^n due to Corollary 1 (again, applied twice). Continue like this d times, ending up with a tree of depth d having 2^d leaves. The time complexity is $d2^n$.

As mentioned, with 2^d leaves, meaning 2^d new target images, finding by brute force a true preimage has complexity 2^{2n-d}. Hence, the total time complexity is about $d2^n + 2^{2n-d}$. Memory requirements are $2^{d+1} - 1$ intermediate hash values and a negligible number of message blocks.

Observe that with $d = 0$ one gets time complexity 2^{2n} and space complexity 1, which is not surprising since we do not build a tree at all, so we have a standard brute force preimage attack. With $d = n/2$ one gets time complexity about $2^{3n/2}$ and space complexity about $2^{n/2}$, equivalent to the attack of Lai and Massey, but the technique is different. The most efficient attack appears when $d = n$, in which case the time complexity is about $(n + 1)2^n$, and the space complexity is 2^{n+1}. We improve the efficiency of this particular time/memory trade-off in Section 4.2.

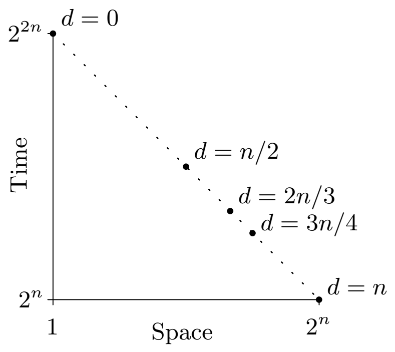

Fig. 5: A visualisation of the time/memory trade-off. Both axes are logarithmic. The case $d = 0$ corresponds to the brute force attack. Larger values of d constitute improvements with respect to attack efficiency.

We note that this attack provides practically any time/memory trade-off for which the product of the time and the space complexities is about 2^{2n}. Figure 5 shows some example trade-offs.

Alternative methods. The tree above does, in fact, not have to be binary. If every node has on average 2^b children, then when the tree has depth d, there are 2^{bd} leaves. The time required to construct the tree is $d2^{b+n-1}$. The time required for Step 2 above is 2^{2n-bd}. The memory requirements are about 2^{bd} for reasonably large b. With $b = n/(d + 1)$, which approximately balances the time spent in Steps 1 and 2, the total time complexity is about $(d/2+1)2^{n(d+2)/(d+1)}$ and the memory requirements are $2^{nd/(d+1)}$.

An alternative way of constructing the tree is the following. First, find a pseudo-preimage of the root. Then, find a pseudo-preimage of the root and its child. Continue applying Corollary 1 this way, finding in each step a pseudo-preimage for each node in the tree, thus doubling the tree size in every step. After d steps, the tree contains 2^d nodes. The time complexity is $d2^{n-1}$. See Figure 6.

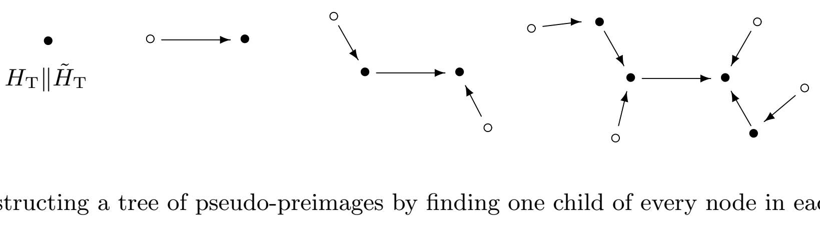

Fig. 6: Constructing a tree of pseudo-preimages by finding one child of every node in each step.

Now, if there is no length padding, then we may perform a brute force search that links the initial value to any of the 2^d nodes in the tree. This brute force search has complexity 2^{2n-d}. Compared to the variant of the previous section, both time and space requirements are roughly halved. We note that this attack resembles a method described by Leurent [17] of finding preimages of MD4.

Length padding can be circumvented in the same way as it is circumvented in Kelsey and Schneier's second preimage attack on the Merkle-Damgård construction [12], but the resulting attack is slightly slower than the variant above, since there is (apparently) no efficient method of finding fixed points of the compression function.

4.2 Pushing the time complexity down to 2^n

The attack above can be modified to obtain an attack of time complexity very close to 2^n. The attack applies a technique which bears some resemblance with the one used in a preimage attack by Mendel and Rijmen on the HAS-V hash function [19], and also with the P^3graph method introduced by De Cannière and Rechberger in [4]. The attack works as follows:

1. Choose two message blocks m_0 and m_1 arbitrarily, but with correct padding for a message of length $n+1$ blocks. Here we assume that padding does not fill an entire message block.
2. Compute $f(i, m_b)$ for each $b \in \{0,1\}$ and for every i from 0 to $2^n - 1$. Store the outputs in the lists U_b, sorted on the output. Sorting can be done in linear time by using, e.g., Bucket-Sort or direct addressing [3].
3. Construct a binary tree with 2^n leaves having the target image $H_{\mathrm{T}} \| \tilde{H}_{\mathrm{T}}$ as root (as above for $d = n$). The two children of each node in the tree are found by lookups in U_0 and U_1, respectively.
4. Given 2^n new target images (namely the leaves in the tree), perform a brute force search starting from the initial value of the hash function.

Step 2 above takes time 2^n. Memory requirements for each of the lists U_b are 2^n values of n bits. Step 3 is expected to take a negligible amount of time compared to Step 2, since the tree is constructed by about 2^n table lookups. Step 4 takes an expected time 2^n, since there are 2^n target images, and the probability of reaching each of them is 2^{-2n}. In total, the time complexity of the attack is about 2^{n+1}, and the memory requirements are about the same.

We note that if padding spans several message blocks, a few initial steps are required to invert through the padding blocks. This may add a small factor of 2^n to the complexity.

Table 2 shows some example complexities of this attack for different sizes of n, compared to the previous best known preimage attack and the brute force attack.

Table 2: Time complexities of the preimage attack of Section 4.2 compared to the previous best known preimage attack of Lai and Massey, and to a brute force attack. For details on the case of DES ($n = 54$), we refer to Appendix A.2.

n	Preimage attack complexity		
	Section 4.2	Lai-Massey	Brute force
54	2^{55}	2^{81}	2^{108}
64	2^{65}	2^{96}	2^{128}
128	2^{129}	2^{192}	2^{256}
256	2^{257}	2^{384}	2^{512}

5 Multicollisions

The preimage attack described in the previous section can be used to construct multicollisions for the construction (2). Let the hash function be H, and let its initial value be $H_{\mathrm{iv}}\|\tilde{H}_{\mathrm{iv}}$. Apply the above preimage attack twice with target hash value $H_{\mathrm{iv}}\|\tilde{H}_{\mathrm{iv}}$, yielding two messages M_0 and M_1. In other words, we find M_0, M_1 such that $H(M_0) = H(M_1) = H_{\mathrm{iv}}\|\tilde{H}_{\mathrm{iv}}$. Now we can construct a 2^t-collision for arbitrary t; the messages in the multicollision consist of t copies of M_0 or M_1, concatenated together.

The time complexity is twice the complexity of the preimage attack, i.e., 2^{n+2}. For $t > 4$ this is more efficient than the previous best known multicollision attack by Joux [11], which has time complexity $t2^n$, assuming a birthday attack is used to produce each individual collision; by applying the collision attack of Section 3, the complexity is reduced to (very roughly) $(t \log_2(n)/n)2^n$. Still the multicollision attack based on the preimage attack is faster when $t > 4n/\log_2(n)$. A drawback of the preimage-based method is memory requirements, which are about 2^{n+1} in our attack, whereas by using cycle-finding methods [2,6], the memory requirements of Joux's attack can be reduced to a negligible quantity.

6 Other non-random properties

Say M is a message of t blocks, and let $H(M) = H_t\|\tilde{H}_t$ be the MDC-2 hash of M. The probability that $H_t \neq \tilde{H}_t$ is $(1 - 2^{-n})^t$, because the two halves must be different after the processing of every block out of the t blocks, in order for them to be different at the end. For an ideal $2n$-bit hash function, this probability is $1 - 2^{-n}$, irrespective of the value of t. Hence, when $t \gg 1$, the probability of the two output halves being equal is much higher in MDC-2 than in an ideal hash function. In fact, if $t = 2^n$, then the probability is around $1 - 1/e \approx 0.63$, since $(1 - 2^{-n})^{2^n} \approx 1/e$ for plausible values of n. The property does not hold for the construction (2) in general (nor does it hold if some key bits are fixed to ensure that the two keys in each iteration are different). What is required is that some n-bit value b exists for every n-bit value a such that $g(a\|a) = b\|b$.

If, during the processing of a message, one has obtained two equal halves, a standard birthday collision attack can be applied in time $2^{n/2}$. Hence, a new type of birthday attack on MDC-2 is as follows. Search for a message block m_0 such that $f(H_0, m_0) = f(\tilde{H}_0, m_0) = H_1$. Then find a pair (m_1, m_1') of message blocks such that $f(H_1, m_1) = f(H_1, m_1')$. This attack takes the same amount of time as a standard birthday attack (it is in fact faster by a factor of two, since f only has to be called 2^n times), but a naive implementation uses only $2^{n/2}$ memory compared to 2^n for a (naive) standard birthday attack. By using cycle-finding methods, memory requirements can be made negligible in both cases.

7 Application to other constructions

The construction (2) can be generalised even further. For example, we may define the following general construction, where f and $\tilde{f}$ are two distinct functions both mapping as $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is (again) an invertible mapping:

$$\begin{aligned} W &= f(H_{i-1}, m_i)\|\tilde{f}(\tilde{H}_{i-1}, m_i) \\ H_i\|\tilde{H}_i &= g(W). \end{aligned} \tag{3}$$

Our attacks also apply to this construction, except that in some cases the complexity is up to twice as high. For instance, finding a pseudo-preimage of $H_{\mathrm{T}}\|\tilde{H}_{\mathrm{T}}$ now requires 2^n evaluations of both f and $\tilde{f}$, and hence the total time complexity is comparable to 2^n compression function evaluations, and not 2^{n-1} as is the case when $f = \tilde{f}$.

Apart from MDC-2 we have not found other constructions in the literature that fall under the category of (2) or (3). However, a construction that easily comes to mind is the dual of MDC-2, meaning that the message block is used as the key in the block cipher calls, and the chaining value is used as the plaintext and also in the feed-forward. An advantage of this dual construction is that some block ciphers accept keys that are larger than the plaintext block, and hence the message blocks are larger which results in improved performance. However, since this construction is an instance of (2), it is susceptible to the attacks described in this paper.

8 Conclusion

In this paper we presented the first collision attack on the MDC-2 construction having time complexity below that of a birthday attack. The attack applies to other constructions similar to MDC-2, and does not rely on weaknesses of the underlying block cipher.

We also described new and improved time/memory trade-offs for preimage attacks, where almost any trade-off such that the product of time and space complexities is about 2^{2n}, with time complexity between 2^n and 2^{2n}, is possible. These new trade-offs mean that, e.g., a second preimage attack on MDC-2 based on DES (see Appendix A) is not far from being practical.

We showed how to construct multicollisions based on the fastest preimage attack, and we discussed some other constructions to which our attacks apply.

We believe the attacks have great theoretical and potential practical significance. Double-length schemes have been studied intensively in the last two or three decades, and for many years it was believed that MDC-2 was collision resistant, assuming the underlying block cipher was secure. In fact, the main criticism of MDC-2 seems to have been its somewhat poor performance. These attacks show that we still have a lot to learn about double-length constructions, although the recent shift towards provably secure schemes provides some consolation.

Acknowledgements

We would like to thank Prof. Bernhard Esslinger of University of Siegen, Joerg-Cornelius Schneider and Henrik Koy of Deutsche Bank for discussions on MDC-2-DES and providing us with references for its use [10, 15, 26]. We also would like to thank the reviewers for insightful comments and for providing the reference to Feller [5] for the estimate (1).

References

1. B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas, Jr., C. H. W. Meyer, J. Oseas, S. Pilpel, and M. Schilling. Data authentication using modification detection codes based on a public one way encryption function, March 13, 1990. US Patent no. 4,908,861. Assigned to IBM. Filed August 28, 1987. See `http://www.google.com/patents?vid=USPAT4908861` (2008/09/02).
2. R. P. Brent. An improved Monte Carlo factorization algorithm. *BIT Numerical Mathematics*, 20(2):176–184, June 1980.
3. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.
4. C. De Cannière and C. Rechberger. Preimages for Reduced SHA-0 and SHA-1. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008, Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 179–202. Springer, 2008.
5. W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 3rd edition, 1968.
6. R. W. Floyd. Nondeterministic Algorithms. *Journal of the Association for Computing Machinery*, 14(4):636–644, October 1967.

7. M. E. Hellman. A Cryptanalytic Time–Memory Trade-Off. *IEEE Transactions on Information Theory*, IT-26(4):401–406, 1980.
8. S. Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In M. J. B. Robshaw, editor, *Fast Software Encryption 2006, Proceedings*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.
9. International Organization for Standardization. ISO/IEC 10118-2:1994. Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher algorithm, 1994. Revised in 2000.
10. International Organization for Standardization. ISO 9735-6:2002. Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules (Syntax version number: 4, Syntax release number: 1) – Part 6: Secure authentication and acknowledgement message (message type – AUTACK), 2002. Available: `http://www.gefeg.com/jswg/v41/data/V41-9735-6.pdf` (2008/09/02).
11. A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. K. Franklin, editor, *Advances in Cryptology – CRYPTO 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004.
12. J. Kelsey and B. Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005.
13. L. R. Knudsen, X. Lai, and B. Preneel. Attacks on Fast Double Block Length Hash Functions. *Journal of Cryptology*, 11(1):59–72, 1998.
14. L. R. Knudsen and B. Preneel. Fast and Secure Hashing Based on Codes. In B. S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO '97, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 485–498. Springer, 1997.
15. D. Kraus. Integrity mechanism in German and international payment systems, 2002. Available: `http://www.src-gmbh.de/whitepapers/Intergrity_mechanisms_in_payment_systems_Kraus_en.pdf` (2008/09/02).
16. X. Lai and J. L. Massey. Hash Functions Based on Block Ciphers. In R. A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT '92, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1993.
17. G. Leurent. MD4 is Not One-Way. In K. Nyberg, editor, *Fast Software Encryption 2008, Proceedings*, volume 5086 of *Lecture Notes in Computer Science*, pages 412–428. Springer, 2008.
18. S. Lucks. A Failure-Friendly Design Principle for Hash Functions. In B. K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer, 2005.
19. F. Mendel and V. Rijmen. Weaknesses in the HAS-V Compression Function. In K.-H. Nam and G. Rhee, editors, *Information Security and Cryptology – ICISC 2007, Proceedings*, volume 4817 of *Lecture Notes in Computer Science*, pages 335–345. Springer, 2007.
20. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
21. C. H. Meyer and M. Schilling. Secure Program Load with Manipulation Detection Code. In *SECURICOM 88, Proceedings*, pages 111–130, 1988.
22. M. Nandi. Towards Optimal Double-Length Hash Functions. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *Progress in Cryptology – INDOCRYPT 2005, Proceedings*, volume 3797 of *Lecture Notes in Computer Science*, pages 77–89. Springer, 2005.
23. M. Nandi, W. Lee, K. Sakurai, and S. Lee. Security Analysis of a 2/3-Rate Double Length Compression Function in the Black-Box Model. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption 2005, Proceedings*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2005.
24. National Bureau of Standards. Data Encryption Standard (DES), Federal Information Processing Standards Publication (FIPS PUB) 46, January 15, 1977.
25. J. P. Steinberger. The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In M. Naor, editor, *Advances in Cryptology – EUROCRYPT 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2007.
26. B. Struif. German Health Professional Card and Security Module Card, Specification, Pharmacist & Physician, v. 2.0, 2003. Available: `http://www.dkgev.de/media/file/2589.spez-engl-3.pdf` (2008/09/02).
27. H. C. A. van Tilborg, editor. *Encyclopedia of Cryptography and Security.* Springer, 2005.
28. J. Viega. The AHASH Mode of Operation, September 2004. Manuscript. Available: `http://www.cryptobarn.com/papers/ahash.pdf` (2008/09/02).

A The special case of MDC-2 instantiated with DES

For simplicity, throughout the paper we assumed that the key size k equals the block size n of the block cipher with which MDC-2 is instantiated. However, this is not necessarily the case, with DES [24] ($n = 64$, $k = 56$) being the most prominent example. The effective key size for MDC-2 with DES is further reduced by two bits to $k = 54$. For the following, it suffices to think of the mapping from chaining blocks to keys as a truncation from 64 to 54 bits. The exact details of this mapping are of no concern for the following treatment, hence we refer to [9] for the full details.

A.1 Collision attacks

The collision attack as described in Section 3 produces a collision in the last chaining value of length $2n$. However, if an arbitrary message block is appended to the expected colliding message pair, it suffices to look for a collision in the $2k$ bits that will be used as the key input of DES in the following iteration. Hence, for the collision attack on MDC-2 based on DES having complexity about $2^{51.5}$, instead of two, at least three message blocks are needed.

A.2 Preimage attacks

Also for the preimage attack of Section 4, the target hash is assumed to be of size $2n$. In order to take advantage of a smaller key size k, the last message block needs to be known by the attacker. In this case the time complexity can be as low as 2^{55}; if no first preimage is given then the attack has a complexity of about 2^{65}.