The Collision Intractability of MDC-2 in the Ideal-Cipher Model

John P. Steinberger

Dept. of Mathematics, University of California, Davis, California 95616 USA
jpsteinb@math.ucdavis.edu

Abstract. We provide the first proof of security for MDC-2, the most well-known construction for turning an n-bit blockcipher into a $2n$-bit cryptographic hash function. Our result, which is in the ideal-cipher model, shows that MDC-2, when built from a blockcipher having blocklength and keylength n, has security much better than that delivered by any hash function that has an n-bit output. When the blocklength and keylength are $n = 128$ bits, as with MDC-2 based on AES-128, an adversary that asks fewer than $2^{74.9}$ queries usually cannot find a collision.

Keywords: Collision-resistant hashing, cryptographic hash functions, ideal-cipher model, MDC-2.

1 Introduction

Overview. A double block length hash-function uses an n-bit blockcipher as the building block by which it maps (possibly long) strings to $2n$-bit ones. The classical double block length hash-function is MDC-2, illustrated in Figure 1. This nearly 20-year-old technique [5, 22] is specified in the ANSI X9.31 and ISO/IEC 10118-2 standards [1, 13], and it is implemented in popular libraries and toolkits, such as OpenSSL.

This paper gives the first proof of security for MDC-2. Our result establishes that when MDC-2 is based on an ideal blockcipher with keylength and blocklength of n bits, the adversary must ask well over $2^{n/2}$ queries to find a collision. In particular, for $n = 128$, no adversry can find a collision with so much as a 50% chance if it asks fewer than $2^{74.9}$ forward-or-backward queries of a 128-bit blackbox-modeled blockcipher.

Getting a collision-resistance bound of $2^{74.9}$ queries when $n = 128$ is still far from the optimum one might hope for, which is a bound of 2^{128} queries for an output of $2n = 256$ bits (the birthday bound). But obtaining *any* bound above 2^{64} (a trivial lower bound) has proved elusive to researchers thus far, given the combinatorial complexity of the problem.

What is MDC-2? Traditionally, MDC-2 was instantiated using DES, and some people may understand MDC-2 to *mean* MDC-2 based on DES. This is not our meaning. Indeed this paper assumes a common keylength and blocklength n bits, and so our results don't directly apply to MDC-2 based on DES. (We assume that, with signficant work, one could extend our analysis to handle the

Algorithm $\mathrm{MDC2}^E(X)$

$X_1 \cdots X_m \leftarrow X$ where $|X_i| = n$

for $i \leftarrow 1$ **to** m **do**

$\quad V_i \leftarrow X_i \oplus E_{A_i}(X_i)$

$\quad W_i \leftarrow X_i \oplus E_{B_i}(X_i)$

$\quad (A_{i+1}, B_{i+1}) \leftarrow (V_i^L W_i^R, W_i^L V_i^R)$

return $V_m W_m$

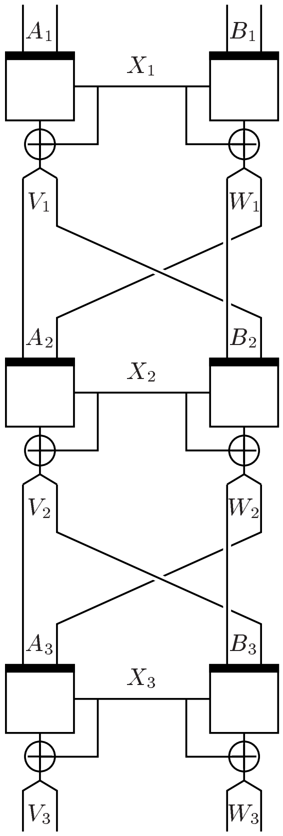


Fig. 1: **Left:** Definition of the MDC-2 algorithm based on a blockcipher E with key length and block length n. The message being acted on is $X = X_1 \cdots X_m$ where $m \geq 1$ and $|X_i| = n$. Strings A_1 and B_1 are distinct n-bit constants. For an even-length sting S we let S^L and S^R be its left and right half. **Right:** Illustration of the algorithm acting on a three-block messsage $X = X_1 X_2 X_3$. The resulting hash is $H(X) = V_3\, W_3$. The darkened edge of the box representing the blockcipher indicates the input that is the key.

DES parameters of 56-bit keys and 64-bit blocks, but we haven't done this.) In this paper we consider MDC-2 using a blockcipher $E\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ with equal-length blocks and keys. We make this assumption for simplicity, while preserving contemporary applicability: eliminating "bit-dropping" makes the algorithm cleaner, while the usage of MDC-2 that people nowadays envisage is with the blockcipher AES-128 [30]. All future mention of MDC-2 in this paper assumes equal blocklength and keylength.

The MDC-2 algorithm is simple and elegant: building on the usual Merkle-Damgård approach [6, 21], the compression function uses two parallel invocations of the Matyas-Meyer-Oseas compression function [20] and then swaps the right halves of the outputs. It is defined and illustrated in Figure 1. It is easy to see that the algorithm doesn't work (that is, it admits efficient attacks) if it is "over-simplified" by dropping the left/right swapping, the feed-forward XOR, or both.

The version of MDC-2 that we consider does not incorporate a "bit fixing" step like replacing the leftmost bit of each left-column blockcipher key in Figure 1 with a 0-bit and replacing the leftmost bit of each right-column blockcipher key with a 1-bit. Such bit-fixing was employed in MDC2-DES [1, 13] to overcome the key-complementation property of the primitive and also, conceivably, as a security measure.

We also comment that in the version of MDC-2 that we consider, no length-annotation or padding is used, and the domain is correspondingly restricted to $(\{0,1\}^n)^+$. It is easy and customary to use padding and length-annotation to extend MDC-2 to handle a domain of any string of less than 2^n bits. Provable-security results immediately extend: a collision-intractability result for the $(\{0,1\}^n)^+$ domain version of a hash function will always lift to give essentially the same bound for the $\{0,1\}^{<n}$ domain version one gets after padding and length annotation.

Our results. We work in the ideal-cipher model, as in [4, 8, 15]. This is the customary model for proving the security of a blockcipher-based hash function. In the ideal-cipher model the underlying primitive, a blockcipher E, is modeled as a family of random permutations $\{E_K\}$ with a random permutation chosen independently for each key K. The adversary may make a query $E_K(X)$ to discover the corresponding value $Y = E_K(X)$, or the adversary may make a query $E_K^{-1}(Y)$ so as to learn the corresponding value $X = E_K^{-1}(Y)$ for which $E_K(X) = Y$. We are interested in the chance that an adversary can find a collision, namely a pair of distinct messages that collide under $\mathrm{MDC2}^E$, by asking q queries. More formal definitions will be given below.

It is easy to show that finding a collision for MDC2 implies finding K, X, K', X' with $(K, X) \neq (K', X')$ such that $E_K(X) \oplus X = E_K(X') \oplus X'$. From this it easily follows (see [4]) that an adversary's chance of finding a collision in q queries is at most $q(q+1)/2^n \approx q^2/2^n$ where $n = |X| = |K|$ is the block size. This is a trivial upper bound, only as good as the conventional bound one expects for a hash function with n-bit output.

Ideally one would like to prove a bound of $q^2/2^{2n}$ for MDC-2, the bound corresponding to the birthday attack, since the output length of MDC-2 is $2n$. However, despite the lack of known attacks on MDC-2, no one has even been able to exhibit an improvement on the trivial bound of $q^2/2^n$. In this paper we give the first improvement by showing that an adversary has chance $O(q^5/2^{3n})$ of finding an attack and therefore needs at least $q \approx 2^{3n/5}$ queries to have an even chance of finding a collision. For example when $n = 128$ (the main case of interest) we show that an adversary needs $q = 2^{74.9}$ queries to have an even chance of obtaining a collision, which is over 2^{10} greater than the trivial bound of $2^{63.5}$. Figure 2 shows our upper bound as function of q for the case $n = 128$.

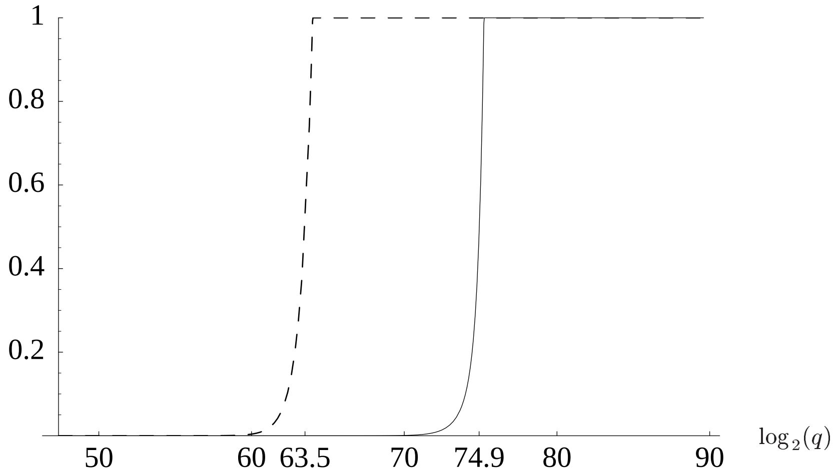


Fig. 2: Our upper bound on $\mathsf{Adv}_{128}^{\mathrm{MDC2}}(q)$ as a function of q (solid line) compared to the previous best upper bound of $q(q+1)/2^{128}$ (dotted line).

2 Preliminaries

Let $\mathsf{Bloc}(n)$ be the set of functions E: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that $E(K,\cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$. Given a blockcipher $E \in \mathsf{Bloc}(n)$ we define $\mathrm{MDC2}^E$: $(\{0,1\}^n)^+ \to \{0,1\}^{2n}$ by the algorithm of Fig. 1. The hash of a word X where $|X|$ is a multiple of n by $\mathrm{MDC2}^E$ is denoted by $\mathrm{MDC2}^E(X)$.

An adversary is a computationally unbounded but always-halting algorithm A with access to an oracle $E \in \mathsf{Bloc}(n)$. We can assume (by standard arguments) that A is deterministic. The adversary can make either a "forward" query $(K_i, X_i)_{\mathrm{fwd}}$ to its oracle E or a "backward" query $(K_i, Y_i)_{\mathrm{bwd}}$. The forward query is answered by $Y_i = E_{K_i}(X_i)$ and the backward query is answered by $X_i = E^{-1}_{K_i}(Y_i)$. Either way the result of the query is stored in a triple (X_i, K_i, Y_i) and the *query history* of A^E, denoted $\mathcal{Q} = \mathcal{Q}(A^E)$, is the tuple $(Q_1, \ldots, Q_q)$ where $Q_i = (X_i, K_i, Y_i)$ is the result of the i-th query made by the adversary, and where q is the total number of queries made by the adversary. If (X_i, K_i, Y_i) is an element of the query history then we refer to X_i as the "word input" of the query, to K_i as the "key" of the query, and to Y_i as the "output" of the query. The quantity $X_i \oplus Y_i$ is called the "XOR output" of the query.

The adversary's goal is to output a pair of nonempty strings X, X' such that $X \neq X'$ and $\mathrm{MDC2}^E(X) = \mathrm{MDC2}^E(X')$. Moreover we impose the condition that the adversary must have made all queries necessary to compute $\mathrm{MDC2}^E(X)$ and $\mathrm{MDC2}^E(X')$. This restriction is reasonable since otherwise the adversary can output very long words X, X' where $\mathrm{MDC2}^E(X) = \mathrm{MDC2}^E(X')$ with good probability but where computing $\mathrm{MDC2}^E(X)$, $\mathrm{MDC2}^E(X')$ is infeasible. (For example, without making any queries, the adversary could simply output

0^{Kn} and 0^{2Kn} where K is the lcm of all numbers between 1 and 2^n and have probability 1 of obtaining a collision, but this isn't a reasonable type of attack.)

Since we may tell simply from the adversary's query history $\mathcal{Q}$ whether it is possible for the adversary to output words $X \neq X'$ such that $\mathrm{MDC2}^E(X) = \mathrm{MDC2}^E(X')$ and such that $\mathcal{Q}$ contains all the queries necessary for the computation of $\mathrm{MDC2}^E(X)$, $\mathrm{MDC2}^E(X')$, we will in fact dispense the adversary from having to output X, X' and simply determine whether the adversary has been successful or not by examining its query history $\mathcal{Q}$. Formally, we say that $\mathsf{Coll}^E(\mathcal{Q})$ holds if there are two distinct nonempty words X, X' of lengths divisible by n such that $\mathrm{MDC2}^E(X) = \mathrm{MDC2}^E(X')$ and such that $\mathcal{Q}$ contains all the queries necessary to compute $\mathrm{MDC2}^E(X)$, $\mathrm{MDC2}^E(X')$ as defined by the algorithm of Fig. 1. The goal of the adversary A is thus to make some sequence of queries $\mathcal{Q} = \mathcal{Q}(A)$ such that $\mathsf{Coll}^E(\mathcal{Q})$. We define the adversary's ability to break MDC-2 by

$$\mathsf{Adv}_n^{\mathrm{MDC2}}(A) = \Pr[E \xleftarrow{\$} \mathsf{Bloc}(n); \mathcal{Q} \leftarrow A^E : \mathsf{Coll}^E(\mathcal{Q})].$$

We let $\mathsf{Adv}_n^{\mathrm{MDC2}}(q)$ be the max over all adversaries A making at most q queries of $\mathsf{Adv}_n^{\mathrm{MDC2}}(A)$. Our goal is thus to upper bound $\mathsf{Adv}_n^{\mathrm{MDC2}}(q)$. We can assume without loss of generality that A always asks exactly q queries and thus that $|\mathcal{Q}(A^E)| = q$.

Say that numbers n and q have been fixed as well as an adversary A such that $|\mathcal{Q}(A^E)| = q$ for all $E \in \mathsf{Bloc}(n)$. If P is any predicate that can be true or false for a sequence of queries $\mathcal{Q}$ (such as $\mathsf{Coll}^E(\mathcal{Q})$) then we write $\Pr[\mathsf{P}(\mathcal{Q})]$ as a shorthand for $\Pr[E \xleftarrow{\$} \mathsf{Bloc}(n); \mathcal{Q} \leftarrow A^E : \mathsf{P}(\mathcal{Q})]$. With this notation we have $\mathsf{Adv}_n^{\mathrm{MDC2}}(A) = \Pr[\mathsf{Coll}^E(\mathcal{Q})]$. We will often use this simpler notation to avoid over-complicating our formulas.

3 Our Security Bound

Our upper bound can be stated in varying degrees of generality and comprehensibility. The most general and least comprehensible statement of our upper bound is the following:

Theorem 1. *Let n, q be natural numbers with $q < 2^n$. Let $N = 2^n$, $N' = N - q$ and let m_a, m_b, m_c be any positive numbers with $eqN^{\frac{1}{2}}/N' \leq m_b \leq N^{\frac{1}{2}}$, $eq/N' \leq m_c$. Finally let $M_b = m_b N'/qN^{\frac{1}{2}}$, $M_c = m_c N'/q$ and $N'' = N'(N^{\frac{1}{2}} - m_b)/N^{\frac{1}{2}}$. Then*

$$\begin{aligned}
&\mathsf{Adv}_n^{\mathrm{MDC2}}(q) \;\leq \\
&q^2/m_a N' + 2qN^{\frac{1}{2}} e^{qN^{\frac{1}{2}} M_b(1-\ln(M_b))/N'} + qNe^{qM_c(1-\ln(M_c))/N'} + && (1)\\
&q(m_a^2 + m_a m_b^2 + m_b^4)/N' + && (2)\\
&q(4m_a m_b)/N' + q(2m_a m_b)/N'' + && (3)\\
&q(m_b^2 m_c + 5m_b^2 + m_a m_c + 6m_a)/N' + q(4m_a + 8m_b^2)/N'' + && (4)\\
&q(4 + 10m_b + 2m_b m_c)/N'' + 3q/N' + 4q/N'' + q^2/N'^2 \quad \blacksquare && (5)
\end{aligned}$$

q	$\mathsf{Adv}_{128}^{\mathrm{MDC2}}(q) \le$	m_a	m_b	m_c
2^{64}	7.57×10^{-7}	2.64×10^6	44.01	3.7147
$2^{68.22}$	10^{-4}	7.01×10^6	128.09	3.9448
$2^{72.19}$	1/100	1.75×10^7	898.95	4.1899
$2^{74.00}$	1/10	2.66×10^7	2902.32	4.3082
$2^{74.72}$	1/3	3.14×10^7	4687.89	4.3523
$2^{74.91}$	1/2	3.29×10^7	5355.49	4.3640
$2^{75.21}$	1	–	–	–

Fig. 3: Upper bounds on $\mathsf{Adv}_{128}^{\mathrm{MDC2}}(q)$ given by Theorem 1. The right three columns specify the values m_a, m_b, and m_c used to obtain the bound of the second column.

For Theorem 1 to give a good bound one must choose suitable values for the constants m_a, m_b, m_c. Choosing large values of m_a, m_b, m_c reduces the terms of line (1) but increases the terms of lines (2)-(5). Unfortunately there is no good closed form for the optimal values of m_a, m_b, m_c (these will change with every q), hence the complex-looking form of Theorem 1. The meaning of the constants m_a, m_b, m_c is explained in the proof.

What Theorem 1 concretely means for $n = 128$ is shown in Figs. 2–3. Fig. 3 shows specific numerical upper bounds for $\mathsf{Adv}_{128}^{\mathrm{MDC2}}(q)$ for various values of q. The threshold value where Theorem 1 gives an upper bound of 1/2 is $q = 2^{74.91}$ (to be compared with the previous best threshold of $q = 2^{63.5}$). For each value of q we also show the values of m_a, m_b, m_c which yield the stated upper bound. Fig. 2 plots our upper bounds on $\mathsf{Adv}_{128}^{\mathrm{MDC2}}(q)$ as a function of q, compared to the previous upper bound of $q(q+1)/N$. The method for optimizing m_a, m_b, m_c for given values of n, q in order to obtain the best bound on $\mathsf{Adv}_{n}^{\mathrm{MDC2}}(q)$ is discussed in the full version of this paper [29]. There we also show (via straightforward calculus) that Theorem 1 implies the following:

Theorem 2. *Let* $q = 2^{\frac{3}{5}n-\epsilon}$ *where* $\epsilon > 0$. *Then* $\mathsf{Adv}_{n}^{\mathrm{MDC2}}(q) \to 0$ *as* $n \to \infty$. ∎

Asymptotically as $n \to \infty$, thus, our bound for $\mathsf{Adv}_{n}^{\mathrm{MDC2}}(q)$ behaves like the function $\min(1, q^5/2^{3n})$, though the two functions still look significantly different for $n = 128$ (e.g. $q^5/2^{3n}$ has a threshold of $2^{76.6}$ for $n = 128$ whereas our bound on $\mathbf{Adv}_{128}^{\mathrm{MDC2}}(q)$ has a threshold of $2^{74.9}$). Though the two functions converge asymptotically there does not seem to be any good closed form relating our bound on $\mathsf{Adv}_{n}^{\mathrm{MDC2}}(q)$ to the function $q^5/2^{3n}$.

4 Analysis

Overview. Rather than analyzing the probability that the queries $\mathcal{Q}$ made by the adversary contain the means of constructing a collision we simplify the problem by analyzing the probability that the queries $\mathcal{Q}$ contain the means of constructing the last two rounds of a collision. Effectively we look to see whether there exist keys K_0, K_1, K_0', K_1' and n-bit words X_1, X_2, X_1', X_2' such that the MDC-2 hash of X_1X_2 using the incoming keys K_0, K_1 (rather than A_1, B_1) equals the MDC-2 hash of $X_1'X_2'$ using the incoming keys K_0', K_1', and such that $\mathcal{Q}$ contains all the queries necessary to make both hashes. Naturally a collision does not necessarily involve two words of at least two blocks each, as either or both words may consist of a single block, and our analysis also allows for this contingency.

To upper bound the probability of the adversary obtaining queries that can be used to construct the last two rounds (or fewer) of a collision we upper bound the probability of the adversary making a query that can be used as the final query to complete such last two rounds. Namely for each i, $1 \le i \le q$, we upper bound the probability that the answer to the adversary's i-th query $(K_i, X_i)_{\mathrm{fwd}}$ or $(K_i, Y_i)_{\mathrm{bwd}}$ (depending) will allow the adversary to use the i-th query to complete (what looks like) the last two rounds of a collision. In the latter case we say the i-th query is "successful", and we give the attack to the adversary.

Naturally this probability will depend on the adversary's first $i-1$ queries. In particular we need to make sure that the adversary hasn't already been too "lucky" with its first $i-1$ queries, or else the probability of the i-th query being successful will be hard to upper bound. An example of being "lucky" would be if there exists a large subset of the first $i-1$ queries that all have the same XOR output (there are two more ways of being lucky defined below). Our upper bound thus breaks down into two pieces: an upper bound for the probability of the adversary getting lucky in one of three specific ways defined below, and the probability of the adversary ever making a successful i-th query, conditioned on the fact that the adversary has not yet become lucky by its $(i-1)$-th query.

Details. Fix numbers n, q and an adversary A asking q queries to its oracle. We upper bound $\Pr[\mathsf{Coll}^E(\mathcal{Q})]$ by exhibiting predicates $\mathsf{Win0}(\mathcal{Q})$, …, $\mathsf{Win8}(\mathcal{Q})$ such that $\mathsf{Coll}^E(\mathcal{Q}) \implies \mathsf{Win0}(\mathcal{Q}) \vee \ldots \vee \mathsf{Win8}(\mathcal{Q})$ and then by upper bounding separately the probabilities $\Pr[\mathsf{Win0}(\mathcal{Q})]$, …, $\Pr[\mathsf{Win8}(\mathcal{Q})]$. Obviously $\Pr[\mathsf{Coll}^E(\mathcal{Q})] \le \Pr[\mathsf{Win0}(\mathcal{Q})] + \cdots + \Pr[\mathsf{Win8}(\mathcal{Q})]$. (The event $\mathsf{Win0}(\mathcal{Q})$ happens if the adversary is lucky, whereas if the adversary is not lucky but makes a successful i-th query then one of the predicates $\mathsf{Win1}(\mathcal{Q})$, …, $\mathsf{Win8}(\mathcal{Q})$ will hold.)

To state the predicates $\mathsf{Win0}(\mathcal{Q})$, …, $\mathsf{Win8}(\mathcal{Q})$ we need some extra definitions. Define functions a, b, b^L, b^R and c on query sequences of length q as follows:

$a(\mathcal{Q}) = |\{(i,j) \in [1 \ldots q]^2 : i \ne j, X_i \oplus Y_i = X_j \oplus Y_j\}|$ is the number of ordered pairs of distinct queries in $\mathcal{Q}$ with same XOR outputs

$b^L(\mathcal{Q}) = \max_{Y \in \{0,1\}^{n/2}} |\{i : (X_i \oplus Y_i)^L = Y\}|$ is the maximum size of a set of queries in $\mathcal{Q}$ whose XOR outputs all have the same left $n/2$ bits

$b^R(\mathcal{Q}) = \max_{Y \in \{0,1\}^{n/2}} |\{i : (X_i \oplus Y_i)^R = Y\}|$ is the maximum size of a set of queries in $\mathcal{Q}$ whose XOR outputs all have the same right $n/2$ bits

$b(\mathcal{Q}) = \max(b^L(\mathcal{Q}), b^R(\mathcal{Q}))$

$c(\mathcal{Q}) = \max_{Y \in \{0,1\}^n} |\{i : X_i \oplus Y_i = Y\}|$ is the maximum size of a set of queries in $\mathcal{Q}$ whose XOR outputs are all the same

The event $\mathsf{Win0}(\mathcal{Q})$ is simply defined by

$$\mathsf{Win0}(\mathcal{Q}) = (a(\mathcal{Q}) \geq m_a) \vee (b(\mathcal{Q}) \geq m_b) \vee (c(\mathcal{Q}) \geq m_c)$$

where m_a, m_b, m_c are the constants from Theorem 1. Thus as m_a, m_b, m_c are chosen larger $\Pr[\mathsf{Win0}(\mathcal{Q})]$ diminishes.

The events $\mathsf{Win1}(\mathcal{Q})$, ..., $\mathsf{Win8}(\mathcal{Q})$ are different in nature from the event $\mathsf{Win0}(\mathcal{Q})$; they concern the feasibility of fitting certain subconfigurations of MDC-2 using queries from $\mathcal{Q} = (X_1, K_1, Y_1), \ldots, (X_q, K_q, Y_q)$. Take for example the configuration $1a$ of Fig. 5. In this configuration, the two strings marked A are equal and the queries marked i, $!i$ are different. These are the only constraints; unmarked strings may or may not be equal, and other queries in the diagram may or may not be equal. Since the bottom left and bottom right queries are distinct fitting the diagram means using two distinct queries $Q_i = (X_i, K_i, Y_i)$ and $Q_{i'} = (X_{i'}, K_{i'}, Y_{i'})$ from $\mathcal{Q}$ for these two positions. We say that four queries $Q_i = (X_i, K_i, Y_i), Q_{i'} = (X_{i'}, Y_{i'}, Y_{i'}), Q_j = (X_j, K_j, Y_j), Q_k = (X_k, K_k, Y_k)$ in $\mathcal{Q}$ "fit" configuration $1a$ if $i \neq i'$ and if Q_i, $Q_{i'}$, Q_j, Q_k can be placed in respectively the bottom left, bottom right, top left and top right positions of configuration $1a$ such that the wiring constraints of the diagram are respected and such that the two strings marked A are equal. Formally, the four queries Q_i, $Q_{i'}$, Q_j, Q_k fit configuration $1a$ if and only if

$$\begin{aligned}
&(i \neq i') \wedge (X_i = X_{i'}) \wedge (X_j = X_k) \wedge (X_i \oplus Y_i = X_{i'} \oplus Y_{i'}) \wedge \\
&((X_j \oplus Y_j)^L = K_i^L) \wedge ((X_j \oplus Y_j)^R = K_{i'}^R) \wedge \\
&((X_k \oplus Y_k)^L = K_{i'}^L) \wedge ((X_k \oplus Y_k)^R = K_i^R).
\end{aligned}$$

Moreover we say that $\mathsf{ExistsFit}_{1a}(\mathcal{Q})$ holds if there exist $i, i', j, k \in [1..q]$ such that queries Q_i, $Q_{i'}$, Q_j, Q_k fit configuration $1a$. The predicates $\mathsf{ExistsFit}_{1b}$, $\mathsf{ExistsFit}_2$, $\mathsf{ExistsFit}_3$, $\mathsf{ExistsFit}_{4a}$, $\mathsf{ExistsFit}_{4b}$, $\mathsf{ExistsFit}_{6a}$, $\mathsf{ExistsFit}_{6b}$, $\mathsf{ExistsFit}_{6c}$, $\mathsf{ExistsFit}_{6d}$, $\mathsf{ExistsFit}_{7a}$, $\mathsf{ExistsFit}_{7b}$, whose configurations are shown in Figs. 5–6, are likewise defined. In these configurations strings marked by the same letter must be equal but strings marked with different letters may or may not be equal; likewise queries marked i, $!i$ or j, $!j$ are different but two queries marked with different letters may be the same. We also let $\mathsf{ExistsFit}_1 = \mathsf{ExistsFit}_{1a} \vee \mathsf{ExistsFit}_{1b}$, $\mathsf{ExistsFit}_4 = \mathsf{ExistsFit}_{4a} \vee \mathsf{ExistsFit}_{4b}$, and so on. Note that $\mathsf{ExistsFit}_{6a} = \mathsf{ExistsFit}_{6b}$ and that $\mathsf{ExistsFit}_{6c} = \mathsf{ExistsFit}_{6d}$, thus $\mathsf{ExistsFit}_6 = \mathsf{ExistsFit}_{6a} \vee \mathsf{ExistsFit}_{6c}$ (configurations $6b$, $6d$ are only provided to facilitate referencing).

Some additional notation is required to indicate inequality between queries in configurations 5 and 8. In these configurations, pairs of queries from the bottom row that do not both contain a '1' or both contain a '0' (namely, queries with different labels) are presumed different; there are no constraints relating top row to bottom row queries, and queries with the same label are not presumed equal (see Fig. 4 for an explanation of "top row", "bottom row"). The predicates $\mathsf{ExistsFit}_5(\mathcal{Q})$, $\mathsf{ExistsFit}_8(\mathcal{Q})$ then denote the existence of a set of queries in $\mathcal{Q}$ fitting respectively configurations 5 and 8 under these constraints.

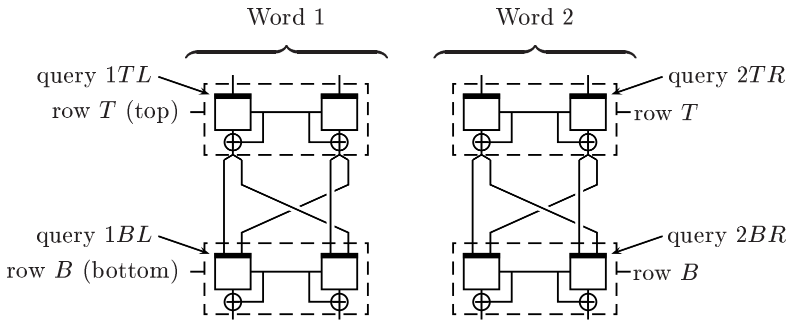


Fig. 4: The query labels.

Let $\mathsf{NotWin}\,j = \overline{\mathsf{Win0}(\mathcal{Q}) \vee \cdots \vee \mathsf{Win}\,j(\mathcal{Q})}$ for $1 \leq j < 8$. We now define:

$$\begin{aligned}\mathsf{Win1}(\mathcal{Q}) &= \mathsf{NotWin0}(\mathcal{Q}) \wedge \mathsf{ExistsFit}_1(\mathcal{Q})\\ \mathsf{Win2}(\mathcal{Q}) &= \mathsf{NotWin1}(\mathcal{Q}) \wedge \mathsf{ExistsFit}_2(\mathcal{Q})\\ &\vdots\end{aligned}$$

and so forth. Thus $\mathsf{Win4}(\mathcal{Q})$, for example, is the predicate which is true if and only if $a(\mathcal{Q}) < m_a, b(\mathcal{Q}) < m_b$, $c(\mathcal{Q}) < m_c$ (these conditions being $\mathsf{NotWin0}(\mathcal{Q})$) and $\mathcal{Q}$ contains queries that fit configurations $4a$ or $4b$ but $\mathcal{Q}$ does not contain queries fitting configurations $1a$, $1b$, 2 or 3.

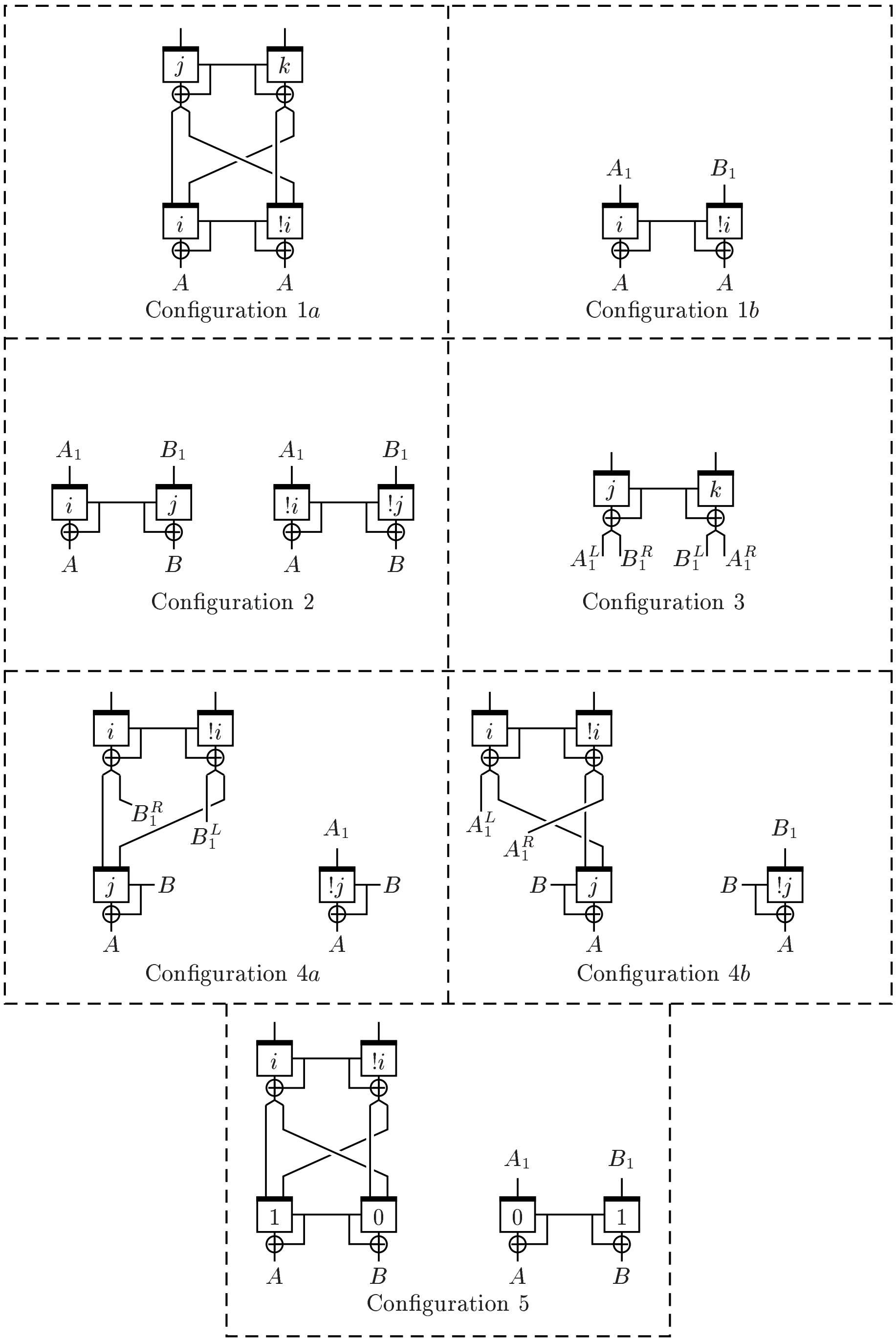
The reader will note that all configurations in Figs. 5–6 have at most two pieces and each piece is a subportion of two rounds of MDC-2. If the configuration has two pieces (such as configurations 2, $4a$, $4b$, 5, $6a$, $6b$, $6c$, $6d$, $7a$, $7b$, 8 as opposed to configurations $1a$, $1b$, 3) then the left portion of the configuration is called "Word 1" and the right portion of the configuration is called "Word 2" (Fig. 4). Queries in the right-hand column of a two-round piece are called "right column" queries and queries in the left-hand column of a two-block portion are called "left column" queries. "Top row" and "bottom row" queries are defined the expected way. A query in the configuration is given coordinates $1TR$ for "Word 1, Top row, Right column" or $2BL$ for "Word 2, Bottom row, Left column", etc. If the configuration has only one piece then we drop the prefix "1" or "2" and simply give coordinates TL, TR, etc. for the queries. The reader should refer to Fig. 4.

We now show that $\mathsf{Coll}^E(\mathcal{Q}) \implies \mathsf{Win0}(\mathcal{Q}) \vee \cdots \vee \mathsf{Win8}(\mathcal{Q})$:

Lemma 1. $\mathsf{Coll}^E(\mathcal{Q}) \implies \mathsf{Win0}(\mathcal{Q}) \vee \cdots \vee \mathsf{Win8}(\mathcal{Q})$.

Proof. First note that $\mathsf{ExistsFit}_1(\mathcal{Q}) \vee \cdots \vee \mathsf{ExistsFit}_8(\mathcal{Q}) \implies \mathsf{Win0}(\mathcal{Q}) \vee \cdots \vee \mathsf{Win8}(\mathcal{Q})$, so it is sufficient to show that $\mathsf{Coll}^E(\mathcal{Q}) \implies \mathsf{ExistsFit}_1(\mathcal{Q}) \vee \cdots \vee \mathsf{ExistsFit}_8(\mathcal{Q})$.

Say $\mathsf{Coll}^E(\mathcal{Q})$. Then a collision can be constructed from the queries $\mathcal{Q}$. We can assume that the collision is earliest possible in the sense that one cannot truncate



j k i $!i$ A A
Configuration 1a
A_1 B_1 i $!i$ A A
Configuration 1b
A_1 B_1 i j A B A_1 B_1 $!i$ $!j$ A B
Configuration 2
j k A_1^L B_1^R B_1^L A_1^R
Configuration 3
i $!i$ B_1^R B_1^L j B A A_1 $!j$ B A
Configuration 4a
i $!i$ A_1^L A_1^R B j A B_1 B $!j$ A
Configuration 4b
i $!i$ 1 0 A B A_1 B_1 0 1 A B
Configuration 5

Fig. 5

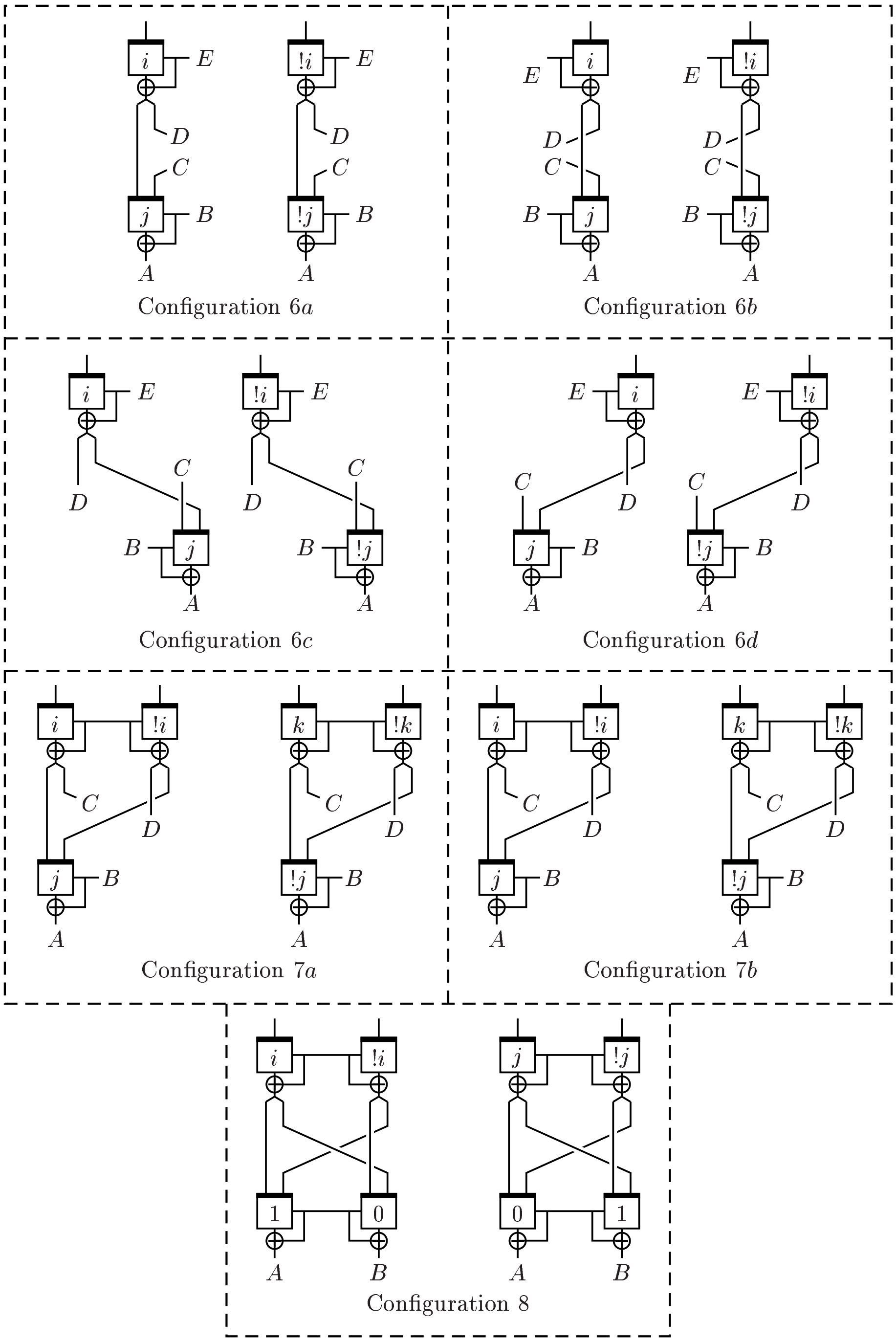

Configuration 6a
Configuration 6b
Configuration 6c
Configuration 6d
Configuration 7a
Configuration 7b
Configuration 8

Fig. 6

either one or both words involved to form a collision from the leftover prefixes (otherwise, take this smaller pair of words). By definition collisions involve words with at least one block, so the collision must either (i) use two words that are one block long each (ii) use one word of at least two blocks and one word of one block or (iii) use two words of at least two blocks each. If the collision uses two words that are one block long each then obviously $\mathsf{ExistsFit}_2(\mathcal{Q})$ (if query i where equal to $!i$, the two words would be the same), so we can assume either (ii) or (iii).

Say first the collision is of type (ii), namely that the collision has one word with $m \geq 2$ blocks, which is WLOG word 1, and one word of with one block, which is word 2. Note first that when word 1 is hashed via MDC-2 there can never be a round where the same query appears both on the left and right-hand sides unless $\mathsf{ExistsFit}_1(\mathcal{Q})$ holds (to see this, take the earliest such round; since the constant keys A_1, B_1 are different this is not the first round and the two queries from the round before are different but have the same XOR output, so $\mathsf{ExistsFit}_1(\mathcal{Q})$). Therefore we can assume that at every round in the hashing of word 1, different queries appear on the left and right-hand sides. Naturally the same query may appear both in the left and right columns in different rounds.

We now examine the last two rounds of the hashing of word 1. The four (not necessarily distinct) queries comprising these two rounds are labeled $1TL$, $1TR$, etc. as in Fig. 4 and as per our convention described above. The two queries making up the unique round for the hashing of word 2 are simply labeled $2L$ and $2R$, where $2L$ is the query with key input A_1 and $2R$ is the query with key input B_1. By our previous remark, queries $1TL$ and $1TR$ are distinct as well as queries $1BL$ and $1BR$. If query $1BL$ equals query $2L$ and query $1BR$ equals query $2R$ then $\mathsf{ExistsFit}_3(\mathcal{Q})$. On the other hand if query $1BL$ is not equal to query $2L$ and query $2BR$ is not equal to query $2R$ then $\mathsf{ExistsFit}_5(\mathcal{Q})$. Therefore we can assume (by symmetry) that query $1BL$ is not equal to query $2L$ but that query $1BR$ equals query $2R$. But then $\mathsf{ExistsFit}_{4a}(\mathcal{Q})$. This concludes the case when the adversary's collision is of type (ii).

We now assume that both of the words involved in the collision have at least two rounds. We examine the last two rounds of the hashing of each word; the queries for these last two rounds are labeled as in Fig. 4. By the same remark as above, the same query cannot appear in both left and right positions at the same round of the same word, so the top row constraints of configuration 8 are satisfied. If query $1BL$ equals $2BL$ and query $1BR$ equals query $2BR$ then the collision is not earliest possible, a contradiction, so we can assume (by symmetry) that query $1BL$ is not equal to query $2BL$. If queries $1BR$ and $2BR$ are equal then $\mathsf{ExistsFit}_{7a}(\mathcal{Q})$ so they too must be unequal. But then $\mathsf{ExistsFit}_8(\mathcal{Q})$ so we are done. $\square$

The reader may have noted that $\mathsf{ExistsFit}_6(\mathcal{Q})$ does not actually appear in the proof of Lemma 1. However $\mathsf{Win6}(\mathcal{Q})$ will be used to upper bound $\Pr[\mathsf{Win7}(\mathcal{Q})]$ (as $\Pr[\mathsf{Win7}(\mathcal{Q})] \leq \Pr[\mathsf{Win6}(\mathcal{Q})] + \Pr[\mathsf{NotWin6}(\mathcal{Q}) \wedge \mathsf{Win7}(\mathcal{Q})]$).

Let $\mathsf{WinFit}(\mathcal{Q}) = \mathsf{Win1}(\mathcal{Q}) \vee \ldots \vee \mathsf{Win8}(\mathcal{Q})$, so $\Pr[\mathsf{Coll}^E(\mathcal{Q})] \leq \Pr[\mathsf{Win0}(\mathcal{Q})] + \Pr[\mathsf{WinFit}(\mathcal{Q})]$. We show:

Lemma 2. *Let N, N', N'', m_a, m_b, M_b, m_c, M_c be as in Theorem 1. Then* $\Pr[\mathsf{Win0}(\mathcal{Q})] \leq q^2/m_a N' + 2qN^{\frac{1}{2}} e^{qN^{\frac{1}{2}} M_b(1-\ln(M_b))/N'} + qNe^{qM_c(1-\ln(M_c))/N'}$. ∎

and:

Lemma 3. *Let N, N', N'', m_a, m_b, m_c be as in Theorem 1. Then:*

$$\begin{aligned}\Pr[\mathsf{WinFit}(\mathcal{Q})] \leq\ & q(m_a^2 + m_a m_b^2 + m_b^4)/N' + \\ & q(4m_a m_b)/N' + q(2m_a m_b)/N'' + \\ & q(m_b^2 m_c + 5m_b^2 + m_a m_c + 6m_a)/N' + q(4m_a + 8m_b^2)/N'' + \\ & q(4 + 10m_b + 2m_b m_c)/N'' + 3q/N' + 4q/N'' + q^2/N'^2.\end{aligned}$$ ∎

Lemmas 2 and 3 imply Theorem 1 (by Lemma 1). The proof of Lemma 2 uses straightforward balls-in-bins probability and can be found in the full version of our paper [29]. The proof of Lemma 3 is more involved and in some sense constitutes the heart of our paper. Here we only give a grief glimpse of the type of analysis involved by showing how to upper bound $\Pr[\mathsf{NotWin0}(\mathcal{Q}) \wedge \mathsf{ExistsFit}_{1a}(\mathcal{Q})]$, which establishes "half" of the upper bound for $\Pr[\mathsf{Win1}(\mathcal{Q}) = \mathsf{NotWin0}(\mathcal{Q}) \wedge (\mathsf{ExistsFit}_{1a}(\mathcal{Q}) \vee \mathsf{ExistsFit}_{1b}(\mathcal{Q}))]$. (Again, the full proof of Lemma 3 is found in the full version.)

For the next proof we use the notational convention that (K_i, X_i) denotes a forward query $(K_i, X_i)_{\mathrm{fwd}}$ and that (K_i, Y_i) denotes a backward query $(K_i, Y_i)_{\mathrm{bwd}}$. The constants N, N', N'' will remain throughout as defined in Theorem 1, namely $N = 2^n$, $N' = N - q$, $N'' = N'(N^{\frac{1}{2}} - m_b)/N^{\frac{1}{2}}$.

Proposition 1. $\Pr[\mathsf{NotWin0}(\mathcal{Q}) \wedge \mathsf{ExistsFit}_{1a}(\mathcal{Q})] \leq q(m_a + m_b^2)/N' + 2qm_b/N''$.

Proof. Let $\mathcal{Q}_i$ denote the first i queries made by the adversary. The term "last query" means the latest query made by the adversary (we examine the adversary's queries (K_i, X_i) or (K_i, Y_i) one at a time, in succession as they come in). The last query is always given index i. We say the last query is "successful" if the output Y_i or X_i for the last query is such that $a(\mathcal{Q}_i) < m_a$, $b(\mathcal{Q}_i) < m_b$, $c(\mathcal{Q}_i) < m_c$ and such that the adversary can use the query (X_i, K_i, Y_i) to fit configuration $1a$ using only queries in $\mathcal{Q}_i$ (in particular, the last query *must* be used in the fitting for that query to count as successful). The goal is thus to upper bound the adversary's chance of ever making a successful last query.

The strategy for upper bounding the probability of the last query being successful is to consider separately the different ways in which the last query can be used to fit the configuration and to upper bound the probability of success in each case, and finally to sum the various upper bounds. For example, the adversary may use the last query only once in the configuration or otherwise in several different positions of the configuration (such as, say, TL and BL). The basic setup for upper bounding the probability of success in a given case is to upper bound the maximum number of different outputs Y_i or X_i (depending on whether the last query is a forward or backward query) that would allow the query (X_i, K_i, Y_i) to be used to fit the configuration, and then to divide this number by $N' = N - q$ (since either Y_i or X_i, depending, is chosen randomly among a set of at least N' different values). That ratio is then multiplied by q, since the adversary makes q queries in all, each of which could become a successful last query.

Case 1: The last query is used exactly once in the configuration. We can assume WLOG that it is used in the left column.

Subcase 1.1: The last query is used in position BL. Say first that the last query is a forward query (K_i, X_i). Since the last query cannot be successful if $b(\mathcal{Q}_{i-1}) \geq m_b$ (by definition) we can assume that $b(\mathcal{Q}_{i-1}) < m_b$. Then since the left half of the XOR output of the query used in position TL must be equal to the left half of K_i there are at most m_b different queries in $\mathcal{Q}_{i-1}$ that could be used in position TL, for the given inputs (K_i, X_i) of the last query. Likewise because the right half of the XOR output of the query used in position TR must be equal to the right half of K_i there are at most m_b different queries in $\mathcal{Q}_{i-1}$ that could be used in position TR. Since X_i together with the outputs of the queries used in positions TL, TR completely determine the query used in position BR, there are therefore at most m_b^2 different queries in $\mathcal{Q}_{i-1}$ which can be used in position BR for the given inputs (K_i, X_i). Therefore there are at most m_b^2 outputs Y_i which would enable the last query be used to fit the configuration at position BL (namely which would enable the XOR output $X_i \oplus Y_i$ of query BL to be equal to the XOR output of query BR), so the chance of success of the last query if it is forward is $\leq m_b^2/N'$.

Now say the last query is a backward query (K_i, Y_i). We cannot reason like for the forward query case that there are only m_b^2 queries in $\mathcal{Q}_{i-1}$ that that can appear in position BR since we do not know the word input X_i anymore. However because the query used in position BR has same XOR output and same word input as the query in position BL it must also have the same output as the query in position BL, which means the output of the query in position BR is actually Y_i. Now because E is a blockcipher, there is exactly at most one possible query for position BR in $\mathcal{Q}_{i-1}$ for any given value of the key of the query in position BR, and since the key can take at most m_b^2 different values (as in the forward case) there are again at most m_b^2 different queries that can be used in position BR. Therefore there are at most m_b^2 different values for X_i which would make the backwards query (K_i, Y_i) successful, so the last query again has chance of success $\leq m_b^2/N'$.

Thus the last query has chance of success $\leq m_b^2/N'$ whether it is a forward or backward query. Multiplying by q, we obtain that the chance of ever making a successful last query of this type is $\leq qm_b^2/N'$. This concludes the analysis of Subcase 1.1.

Note: we will not always give as many details as in Subcase 1.1. In particular, we will not continue to remind that one can assume $a(\mathcal{Q}_{i-1}) < m_a$, $b(\mathcal{Q}_{i-1}) < m_b$, $c(\mathcal{Q})_{i-1} < m_c$ (or else the last query is by definition not successful) and we will often shorten phrases of the type "query used in position TL" to simply "query TL".

Subcase 1.2: The last query is used in position TL. Because the queries use in positions BL, BR are distinct but have the same XOR output there are at most m_a different ordered pairs of queries in $\mathcal{Q}_{i-1}$ that can be used for the pair BL, BR. But the pair of queries for BL, BR completely determines what the XOR output $X_i \oplus Y_i$ of the last query should be. Therefore the last query has chance at most m_a/N' of success and the total probability of making this type of successful last query is $\leq qm_a/N'$.

Note: Subcase 1.2 does not require a separate analysis for the forward and backward case because we can upper bound the maximum number of successful XOR outputs for the last query *without* looking at the inputs for the last query; by contrast, in Subcase 1.1 we inspected X_i in the forward case and Y_i in the backward case in order to determine the maximum possible number of successful XOR outputs. In general, whenever an upper bound on the total number of successful XOR outputs for the last query can be found without inspecting any inputs for the last query besides the key, the same analysis will work both for the forward and backward cases.

Case 2: The last query is used twice or more in the configuration. Because queries BR and BL are distinct the queries TR and TL are also distinct and so the last query must in fact appear exactly twice in the configuration. We can assume WLOG that it is used in position TL.

The type of analysis we use for this case is slightly different than the analysis for Subcases 1.1, 1.2. To estimate the probability of the last query succeeding we will first look at the left $n/2$ bits of XOR output, estimate their probability P_l of success (the left bits are "successful" if they do not preclude the last query from being successful) and then we estimate the probability of success $P_{r|l}$ of the right $n/2$ bits of XOR output being successful, conditioned on the fact that the left $n/2$ bits are successful (the right $n/2$ bits are "successful" if the last query is successful). The probability of success of the last query is then $P_l P_{r|l}$. Note that if the set of left half of XOR outputs which are successful has size T then $P_l \leq TN^{\frac{1}{2}}/N'$ since the return to any query has chance $\leq N^{\frac{1}{2}}/N'$ of having its left half of XOR output equal to any particular value (there are at most $N^{\frac{1}{2}}$ strings that have that left half, each of which is returned with chance at most $1/N'$). Then if the left half is successful and there are U different possible ways of completing the left half into a successful string, namely U different successful right halfs, the chance of the right half being successful given $\mathsf{NotWin0}(\mathcal{Q}_{i-1})$ is $\leq U/(N^{\frac{1}{2}} - m_b)$ since the XOR output could be any of at least $N^{\frac{1}{2}} - m_b$ values with equal probability (there are at most m_b values which we know will not appear because they have already appeared for this left half). So the total chance of success of the last query in this case (assuming U was independent of the left half, as it will be in our analysis) is $\leq TUN^{\frac{1}{2}}/N'(N^{\frac{1}{2}}-m_b)$ or $\leq TU/N''$.

Subcase 2.1: The last query is used in positions TL, BL. Since the last query appears in positions TL, BL the left half of the last query's XOR output must be equal to the left half of its key input, so the left half of output has chance $P_l \leq N^{\frac{1}{2}}/N'$ chances of succeeding. If it succeeds, there are at most m_b queries for BR in $\mathcal{Q}_{i-1}$ with that left half of XOR output (which must be shared with query BL), so the right half of XOR output has chance $P_{r|l} \leq m_b/(N^{\frac{1}{2}} - m_b)$ of succeeding if the the left half succeeds. Therefore the last query has chance $P_l P_{r|l} \leq m_b N^{\frac{1}{2}}/N'(N^{\frac{1}{2}} - m_b) = m_b/N''$ of succeeding and the adversary's total chance of making this kind of successful last query is $\leq q m_b/N''$.

Subcase 2.2: The last query is used in position TL and in position BR. One can apply the same type of analysis as for Subcase 2.1, showing that the total chance of a successful last query of this type is $\leq q m_b/N''$.

Subcase 2.2 concludes Case 2 and thus all possible cases of making a successful query for configuration $1a$. Summing up the probabilities we get that $\Pr[\mathsf{NotWin0}(\mathcal{Q}) \wedge \mathsf{ExistsFit}_{1a}(\mathcal{Q})] \leq q(m_a + m_b^2)/N' + 2qm_b/N''$. $\square$

5 Conclusion

We have proved the first nontrivial security bound for MDC-2. While such a bound has been a long time coming, we expect that our result is only a first foot in the door. In particular there remains a large gap between the best-known collision-finding attack, which is the trivial attack that succeeds with chance $q^2/2^{2n}$, and the security bound of Theorem 1. Likely our security bound is far from optimal, and it remains an interesting open question to find matching upper and lower bounds.

Acknowledgments

This work was supported in part by NSF CCR-0208842 and a gift from Intel Corporation; thanks to Jesse Walker for sponsoring this research. Part of this work was carried out while the author was visiting NTT labs in Yokosuka, Japan; thanks to Tatsuaki Okamoto for his kind support. The research topic was suggested to the author by Phillip Rogaway, who also provided patient mentoring and guidance throughout the project.

References

1. ANSI X9.31. Public key cryptography using reversible algorithms for the financial services industry. American National Standards Institute, 1998.

2. B. den. Boer and A. Bosselaers. Collisions for the compression function of MD5. *Advances in Cryptology – EUROCRYPT '93*, Lecture Notes in Computer Science, vol. 765, Springer, pp. 293–304, 1993.
3. J. Black, M. Cochran, and T. Shrimpton. On the impossibility of highly efficient blockcipher-based hash functions. *Advances in Cryptology – EUROCRYPT '05*, Lecture Notes in Computer Science, vol. 3494, Springer, pp.–546-541, 2005.
4. J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Compuer Science, vol. 2442, Springer, pp. 320–355, 2002.
5. B. Brachtl, D. Coppersmith, M. Hyden, S. Matyas, C. Meyer, J. Oseas, S. Pilpel, and M. Schilling. Data authentication using modification detection codes based on a public one-way encryption function. US Patent #4,908,861. Awarded March 13, 1990 (filed Auguest 28, 1987).
6. I. Damgård. A design principle for hash functions. *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science, vol. 435, Springer, pp. 416–427, 1990.
7. H. Dobbertin. The status of MD5 after a recent attack. *CryptoBytes* 2 (2), 1996.
8. S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. *Advances in Cryptology – ASIACRYPT '91*, Lecture Notes in Computer Science, vol. 739, Springer, pp. 210–224, 1991.
9. M. Hattori, S. Hirose, and S. Yoshida. Analysis of double block lengh hash functions. *Cryptography and Coding, 9th IMA International Conference*, Lecture Notes in Computer Science, vol. 2898, Springer, pp. 290–302, 2003.
10. S. Hirose. Provably secure double-block-length hash functions in a black box model. *Information Security and Cryptology—ISISC '04*, Lecture Notes in Computer Science, vol. 3506, Springer, pp. 330-342, 2005.
11. S. Hirose. Some plausible constructions of double-block-length hash functions. *Fast Software Encryption (FSE '06).* Lecture Notes in Computer Science, vol. 4047, Springer, pp. 210–225, 2005.
12. W. Hohl, X. Lai, T. Meier, and C. Waldvogel. Security of iterated hash functions based on block ciphers. *Advances in Cryptology – CRYPTO '93.* Lecture Notes in Computer Science, vol. 773, Springer, pp. 303–311, 1993.
13. ISO/IEC 10118-2:2000. Information technology – Security techniques – Hash functions – Hash functions using an n-bit block cipher. International Organization for Standardization, Geneva, Switzerland, 2000. First released in 1992.
14. A. Joux. Multicollisions in iterated hash functions, applications to cascaded constructions. *Advances in Cryptology – CRYPTO '04.* Lecture Notes in Computer Science, vol. 3152, Springer, pp. 306–316, 2004.
15. J. Kilian and P. Rogaway. How to Protect DES Against Exhaustive Key Search. *Journal of Cryptology*, vol. 14, no. 1, pp. 17–35, 2001.
16. L. Knudsen, X. Lai, and B. Preneel. Attacks on fast double block length hash functions. *Journal of Cryptology*, vol. 11, no. 1, pp. 59–72, 1998.
17. X. Lai and J. Massey. Hash functions based on block ciphers. *Advances in Cryptology – EUROCRYPT '92.* Lecture Notes in Computer Science, vol. 658, Springer, pp. 55–70, 1992.
18. W. Lee, M. Nandi, P. Sarkar, D. Chang, S. Lee, and K. Sakurai. PGV-style block-cipher-based hash families and black-box analysis. *IEICE Transactions 88-A(1)*, pp. 39–48, 2005.

19. S. Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004.
20. S. Matyas, C. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Bulletin, 27, pp. 5658–5659, 1985.
21. R. Merkle. One way hash functions and DES. *Advances in Cryptology – CRYPTO '89*. Lecture Notes in Computer Science, vol. 435, Springer, pp. 428–446, 1990.
22. C. Meyer and S. Matyas. Secure program load with manipulation detection code. *Proceedings of the 6th Worldwide Congress on Computer and Communications Security and Protection (SECURICOM '88)*, pp. 111–130, 1988.
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. *Fast Software Encryption (FSE '05)*, Lecture Notes in Computer Science, vol. 3557, pp. 243–254, 2005.
24. M. Nandi. Towards optimal double-length hash functions. *Progress in Cryptography – INDOCRYPT '05*, Lecture Notes in Computer Science, vol. 3797, Springer, pp. 77–89, 2005.
25. M. Rabin. Digitalized signatures. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, Academic Press, pp. 155–168, 1978.
26. R. Rivest. The MD4 message digest algorithm. *Advances in Cryptology – CRYPTO '90*, Lecture Notes in Comptuer Science, vol. 537, pp. 303–311, 1991.
27. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision Resistance. *Fast Software Encryption (FSE '04)*, Lecture Notes in Computer Science, vol. 3017, pp. 371-388, Springer, vol. 3017, 2004.
28. T. Satoh, M. Haga, and K. Kurosawa. Towards secure and fast hash functions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82–A No. 1, pp. 55–62.
29. J. Steinberger. The collision intractability of MDC-2 in the ideal-cipher model. Full version of this paper. Cryptology ePrint Archive, Report 2006/294, 2006.
30. J. Viega. The AHASH mode of operation. Manuscript, 2004. Available from www.cryptobarn.com.
31. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. *Advances in Cryptology – EUROCRYPT '05*, Lecture Notes in Computer Science, vol. 3494, Springer, pp. 1–18. 2005.
32. X. Wang, Y. Yin, and H. Yu. Finding collisions in the full SHA-1. *Advances in Cryptology – CRYPTO '05*, Lecture Notes in Computer Science, vol. 3621, Springer, pp. 17–36, 2005.