Security Proofs for the MD6 Hash Function Mode of Operation

by

Christopher Yale Crutchfield

B.S., Electrical Engineering and Computer Science (2006)
B.S., Engineering Mathematics and Statistics (2006)
University of California, Berkeley

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author . . . .
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by . . . .
Ronald L. Rivest
Andrew and Erna Viterbi Professor of
Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Security Proofs for the MD6 Hash Function Mode of Operation

by

Christopher Yale Crutchfield

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the requirements
for the degree of Master of Science
Revised on June 3, 2008

Abstract

In recent years there have been a series of serious and alarming cryptanalytic attacks on several commonly-used hash functions, such as MD4, MD5, SHA-0, and SHA-1 [13, 38]. These culminated with the celebrated work of Wang, Yin, and Yu from 2005, which demonstrated relatively efficient methods for finding collisions in the SHA-1 hash function [37]. Although there are several cryptographic hash functions — such as the SHA-2 family [28] — that have not yet succumbed to such attacks, the U.S. National Institute of Standards and Technology (NIST) put out a call in 2007 for candidate proposals for a new cryptographic hash function family, to be dubbed SHA-3 [29].

Hash functions are algorithms for converting an arbitrarily large input into a fixed-length *message digest*. They typically consist of two main components: a *compression function* that operates on fixed-length pieces of the input, and a *mode of operation* that governs how apply the compression function repeatedly on the pieces in order to allow for arbitrary-length inputs. *Cryptographic* hash functions are furthermore required to have several important and stringent security properties including (but not limited to) first-preimage resistance, second-preimage resistance, collision resistance, pseudorandomness, and unpredictability.

This work presents proofs of security for the mode of operation of the MD6 cryptographic hash function [32] — a candidate for the SHA-3 competition — which differs greatly from the modes of operation of many commonly-used hash functions today (MD4, MD5, as well as the SHA family of hash functions.) In particular, we demonstrate provably that the mode of operation used in MD6 preserves some cryptographic properties of the compression function — that is, assuming some ideal conditions about the compression function used, the overall MD6 hash function is secure as well.

Thesis Supervisor: Ronald L. Rivest
Title: Andrew and Erna Viterbi Professor of
Electrical Engineering and Computer Science

Acknowledgments

I would like to thank Professor Ron Rivest for all of his support and insight in the undertaking of this project, as well as for designing the MD6 hash function itself. In addition, many thanks are owed to Jacob Scott, who tirelessly (and sleeplessly) provided many helpful comments on earlier drafts of this work. Finally, I must acknowledge Pam Satjawatcharaphong, who helped keep me afloat throughout this whole process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Hash functions are a fundamental primitive in the field of cryptography, used widely in a broad spectrum of important applications including: message integrity and authentication [3, 4], digital signatures [9], secure timestamping, and countless others. The security of these applications often rely directly on the security properties of the underlying hash function; if the hash function fails to be as secure as believed, then the application fails to be secure as well. Therefore there is often a strong interest in proving, with the most rigorous scrutiny possible, that a given hash function algorithm indeed has the desired security properties and is resilient to a variety of attacks. In this thesis, we will address some of these concerns for the MD6 hash function and prove that its mode of operation preserves some security properties of its compression function.

A ***hash function*** H is an efficiently-computable algorithm that takes as input an arbitrary-length message M and potentially a fixed-length key K (if we are considering a keyed hash function), and produces a fixed-length output D called the ***message digest***.

$$H(K, M) = D$$

One practical use of hash functions in cryptography is in the so-called *hash-and-sign* paradigm [9] for digital signature schemes. This involves, for a signature scheme

σ and a collision-resistant hash function H, hashing the message and then signing the hash: $\sigma(H(m))$. In this way, the hash function H plays the role of a ***domain extender*** for the underlying signature scheme σ, and the overall unforgeability of the resulting signature scheme is relies heavily on the collision resistance of H. The rationale for such constructions is that it is very difficult to design a signature scheme without a hash function that is secure for inputs of arbitrary length, and that it is significantly easier to design such a scheme when the inputs are assumed to be bounded.

For example, in the RSA signature scheme [33], $\sigma_{\text{RSA}}(m) = m^d \pmod N$, where d is the secret exponent and N is the RSA modulus. If we were to allow inputs of arbitrary lengths, it is clear that this scheme fails to have the property of unforgeability, as $\sigma_{\text{RSA}}(m) \equiv \sigma_{\text{RSA}}(m + N) \pmod N$. That is, given a signature pair $(m, \sigma_{\text{RSA}}(m))$ from a signing oracle, an adversary can trivially forge signature pairs $(m + kN, \sigma_{\text{RSA}}(m))$ for any integer k (and thus produce signatures on messages that the oracle did not sign). However, if instead we first hash the message into $\mathbb{Z}_N^*$ and *then* sign it — symbolically, $\sigma_{\text{RSA}}(H(m))$ — it is clear that in order to forge a signature, the adversary must either break the underlying RSA signature scheme (by, say, factoring the modulus N) or find a collision in the underlying hash function H.

Applications such as the aforementioned hash-and-sign paradigm, or message authentication, secure timestamping, as well as a host of other uses in various cryptographic protocols, often make several assumptions about the underlying hash function (which, when it has some or all of these properties, we call a *cryptographic hash function*). In particular, we may assume that the hash function may have the following properties (we will define them more rigorously later on).

- ***Collision resistance***: An adversary should not be able to find two distinct messages M and M' such that $H(M) = H(M')$ (a collision). As shown in the above hash-and-sign example, the security of the signature scheme using H depends strongly on the collision resistance of H.

- ***First preimage resistance***: An adversary given a target image D should not be able to find a preimage M such that $H(M) = D$. One reason (among many) why this property is important is that on most computer systems user passwords are stored as the cryptographic hash of the password instead of just the plaintext password. Thus an adversary who gains access to the password file cannot use it to then gain access to the system, unless it is able to invert target message digests of the hash function.

- ***Second preimage resistance***: An adversary given a message M should not be able to find another message M' such that $M \neq M'$ and $H(M') = H(M)$. This property is implied by collision resistance.

- ***Pseudorandomness***: (For keyed hash functions) an adversary should not be able to distinguish the outputs of $H(K, \cdot)$ from a truly random function. Note that pseudorandomness necessarily implies unpredictability, meaning a pseudorandom function (PRF) naturally is a message authentication code (MAC). However, the converse is not necessarily true, and indeed PRF is a much stronger condition than unpredictability.

- ***Unpredictability***: (For keyed hash functions) an adversary given oracle access to $H(K, \cdot)$ should not be able to forge the output of a message it did not query. That is it should not be able to produce a hash pair (M, D) where $H(K, M) = D$ without having already queried M. We say that if a function is unpredictable, then it is a message authentication code (MAC).

1.1 Mode of Operation

A ***mode of operation*** $\mathcal{M}$ is an algorithm that, given a fixed-length compression function or block cipher f, describes how to apply f repeatedly on fixed-length chunks of the arbitrarily-sized input in order to produce a fixed-length output for the whole. In this way, one can construct Variable Input Length (VIL) cryptographic primitives from Fixed Input Length (FIL) cryptographic primitives, which is an a functionality commonly referred to as domain extension [16].

1.1.1 Iterative Modes of Operation

Many common hash functions in use today — such as MD5 or SHA-1 — are based on an iterative chaining mode of operation frequently referred to as the Merkle-Damgård construction [15, 26]. The Merkle-Damgård construction typically makes use of a compression function $f : \{0,1\}^{n+\ell} \to \{0,1\}^n$, or a block cipher E made to behave as a compression function via the Davies-Meyer transform [25]: $f(x, y) = E_x(y) \oplus y$.

If f is a compression function as defined above, then the plain Merkle-Damgård construction that uses this compression function, MD^f, begins first by padding the input message m to have a length that is an integer multiple of ℓ, and picking some fixed n-bit initial vector IV. It then proceeds sequentially through the ℓ-bit message chunks, starting from the first chunk, and ending after processing the last one.

Algorithm MD^f

Input: $m = m_1 \| m_2 \| \cdots \| m_t$, where $|m_i| = \ell$ for all i.

Output: The message digest, D.

```
1.  $y_0 \leftarrow IV$
2.  **for** $i \leftarrow 1$ **to** $t$
3.          $y_i \leftarrow f(y_{i-1}, m_i)$
4.  $D \leftarrow y_t$
```

The Merkle-Damgård construction has been well-studied in the literature, and variations on this mode of operation (e.g. *strengthened Merkle-Damgård* and others) have been shown to be domain extenders for various cryptographic properties: collision-resistance [2,10,15,26], pseudorandomness [4,5], unforgeability (MAC) [1,24], indifferentiability from a random oracle [14], and several others.

1.1.2 MD6 Mode of Operation

However, MD6 makes use of a substantially different tree-based mode of operation that allows for greater parallelism [32]. Whereas the Merkle-Damgård construction, when viewed as a graph, is essentially a long chain, MD6 may be viewed as a tree-like construction, with a 4-to-1 compression function reducing the overall length of the message at each level.

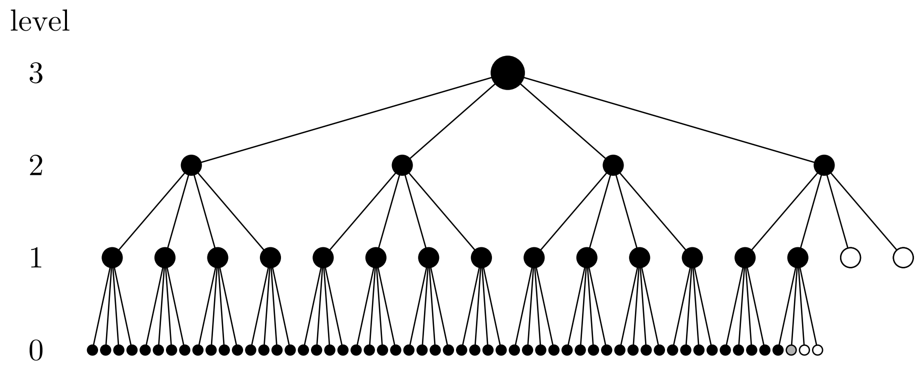

Figure 1-1: The MD6 mode of operation. The computation begins from the bottom and works its way to the top; the root node represents the final compression function which outputs the message digest.

What makes this particular mode of operation different from other tree-based hashing and MAC schemes in the literature [11, 27] is that each node in the tree is labeled with some *auxiliary* information that also feeds into the compression function. In particular, each node is given a unique identifier (effectively changing the characteristic of the compression function at each node in the tree) and the root node is "flagged" with a bit z that identifies that it is the final compression function used. This auxiliary information encoded into the input of the each compression function prevents the type of hash function attacks whereby an adversary can produce a cleverly-constructed message query that corresponds to some substructure of another query (for example, preventing length-extension attacks).

1.2 NIST SHA-3 Competition

Although the SHA-2 family of hash functions has not yet succumbed to the kind of collision-finding attacks that have plagued MD5 and SHA-1 (among others) in recent years, the U.S. National Institute of Standards and Technology put out a call in 2007 for candidate algorithms for a new cryptographic hash function family, called SHA-3. As stated in the call for submissions [29], "a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures", thus necessitating the design of an even more resilient cryptographic hash function family. Although SHA-3 candidates will not differ from SHA-2 in the size of the message digest (which may vary from 224, 256, 384, and 512 bits) or the size of other input parameters such as the key, NIST expects that candidate proposals will improve upon the SHA-2 designs by allowing for randomized (salted) hashing, being inherently parallelizable to take advantage of today's multicore processor design, and being resilient to length extension attacks that many Merkle-Damgård-based hash functions succumb to. On the last two points, "NIST is open to, and encourages, submissions of hash functions that differ from the traditional Merkle-Damgård model, using other structures, chaining modes, and possibly additional inputs."

In terms of the security requirements for a proposed SHA-3 candidate, the call for submissions specifically states the following conditions (this is a subset of the security requirements listed; see [29] for the remainder). For a message digest of d bits, candidate hash functions must have

(1) Collision resistance of approximately $d/2$ bits.

(2) First-preimage resistance of approximately d bits.

(3) Second-preimage resistance of approximately $d - k$ bits for any message shorter than 2^k bits.

For the keyed variant of the candidate hash function proposal, NIST requires that the hash functions supports HMAC (keyed hash function message authentication codes), PRF (pseudorandom function), as well as randomized hashing. An additional security requirement for these modes (for a message digest of d bits) is:

(4) When using HMAC to construct a PRF, the PRF should not be distinguishable from a truly random function with significantly fewer than $2^{d/2}$ queries to the hashing oracle and computation significantly less than a preimage attack.

In this paper we address these security properties, as pertains particularly to the MD6 mode of operation itself. That is, in this paper we take an agnostic approach to the compression function used and consider only a black-box function f with some desirable property P (e.g. collision resistance, pseudorandomness, *et cetera*). Our goal is then to show that the MD6 mode of operation acts as a domain extender for f that preserves the property P. The question we attempt to answer is, does the MD6 mode of operation dilute the security of the black-box function with respect to P, and if so, by how much? For collision resistance, first-preimage resistance, and pseudorandomness we give concrete security bounds for the property preservation. For the MAC functionality, we do the same, although we must introduce a new assumption. For the property of second-preimage resistance, we are unable to demonstrate provably that it preserves this property (although we reduce to a weaker property instead).

1.3 Organization

This thesis is organized as follows. A detailed description of the MD6 mode of operation, basic definitions that will be used throughout the paper, and a short summary of related work will be presented in Chapter 2. In Chapter 3, we will address the collision resistance and preimage resistance of the MD6 mode of operation. Chapter 4 is devoted to topics regarding the pseudorandomness-preserving properties of MD6. In Chapter 5, we demonstrate that the MD6 mode of operation preserves unpredictability as well (under certain assumptions), and therefore is well-suited for use as a MAC. In Chapter 6 we conclude with a summary of results and directions for future work.

There is also an appendix on some supplementary topics. Appendix A discusses the birthday paradox and the birthday bound, which we make use of in a few of our proofs.

Chapter 2

Preliminaries

2.1 The MD6 Cryptographic Hash Function

The MD6 hash function is comprised of two main components: the MD6 compression function and the MD6 mode of operation. The MD6 compression function maps 89 64-bit words of input (64 words of data B, 8 words for the key K, 15 fixed words Q, and 2 auxiliary information words) down to 16 64-bit words of output. Therefore in practice it is a function $f : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^c$ with $k = 8w$, $n = 66w$, and $c = 16w$ (where $w = 64$) (see Figure 2-1).

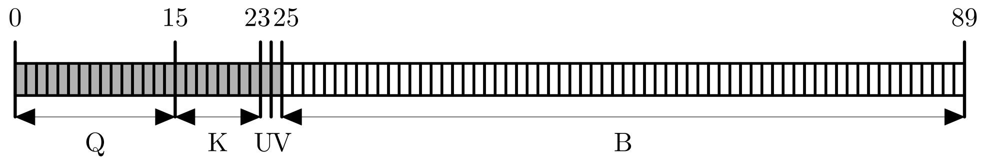

Figure 2-1: The compression function input contains 89 64-bit words: a 15-word constant vector Q, an 8-word key K, a one-word unique node ID U, a one-word control variable V, and a 64-word data block B. The first four items form the auxiliary (or header) information, shown in grey.

Note that although it takes in 89 words of input, 15 words are fixed for the constant Q, hence in practice it is only a function on $74w$ total words of input ($8w$ of which are assigned for the key). In addition, since the data portion of its input is of length $64w$ and its output is $c = 16w$, the MD6 compression function represents a four-fold reduction in the size of data input to the size of the output. Although there is much more to the MD6 compression function than that, this is all that we will assume about its construction. See the MD6 documentation for a much more detailed description of the compression function [32].

As mentioned previously, two words out of the 66 word input space are reserved for control words U and V. A short description of these auxiliary inputs follows.

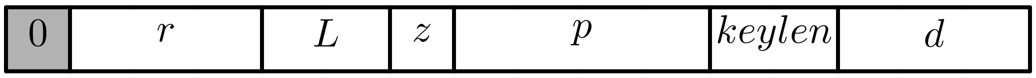

Figure 2-2: Layout of the control word V. The high-order 4 bits are zero (reserved for future use). The 12-bit field r gives the number of rounds in the compression function. The 8-bit field L gives a mode of operation parameter (maximum tree height). The 4-bit field z is 1 iff this compression operation is the very last one. The 16-bit field p gives the number of padding bits in the data block B. The 8-bit field *keylen* gives the number of key bytes in the supplied key K. The 12-bit digest size field d gives the size of the final desired hash function output, in bits.

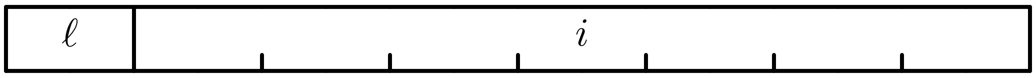

Figure 2-3: Layout of the unique node ID word U. The high-order byte is ℓ, the level number. The seven-byte field i gives the index of the node within the level ($i = 0, 1, \ldots$).

2.1.1 MD6 Mode of Operation

The MD6 mode of operation describes how to apply the fixed-length compression function f repeatedly in order to create a fixed-length digest from an arbitrarily-long input. The standard mode of operation is a hierarchical, tree-based mode of operation. However, MD6 is also parameterized by a "maximum level" parameter L, where $0 \leq L \leq 31$. If $L = 31$, then the MD6 mode of operation is a full 4-ary hash tree. If $L = 0$ then its mode of operation is iterative, similar to Merkle-Damgård. Figures 2-4, 2-6, and 2-5 give a more detailed description of the MD6 mode of operation. In Figures 2-7, 2-8, and 2-9, we give a graphical representation of the MD6 mode of operation.

2.1.2 Notation

Throughout this paper we will reserve the variable names in Table 2.1.2 to describe certain quantities. This is mostly consistent with the MD6 documentation [32, Appendix D], although there are some differences.

Variable	Default	Usage
B	–	the data block portion of a compression function input.
b	64	the number of words in array B.
C	–	the output of the compression function.
c	$16w$	number of bits in the "chaining variable" C.
d	–	number of bits in the MD6 final output ($1 \leq d \leq 512$).
f	–	the MD6 compression function mapping $\{0,1\}^{n+k}$ to $\{0,1\}^{cw}$.
K	0	the key variable (an input to f).
k	$8w$	number of bits in the key variable K.
$keylen$	0	the length in bytes of the supplied key; $0 \leq keylen \leq kw/8$.
ℓ	–	the level number of a compression node.
L	31	mode parameter (maximum level number).
N	–	the non-key, non-Q piece of the compression function input.
n	$66w$	the size of N (in bits).
p	–	the number of padding bits in a data block B.
Q	–	an approximation to $\sqrt{6}$ (see [32, Appendix A]).
q	15	the length of Q in words.
U	–	one-word unique node ID.
u	1	length of U in words.
V	–	a control word input to a compression function.
v	1	length of V in words.
w	64	the number of bits in a word.
z	–	flag bit in V indicating this is final compression.

Table 2.1: Variable naming conventions used throughout this paper.

The MD6 Mode of Operation

Input:

M : A message M of some non-negative length m in bits.
d : The length d (in bits) of the desired hash output, $1 \le d \le 512$.
K : An arbitrary $k = 8$ word "key" value, containing a supplied key of *keylen* bytes.
L : A non-negative mode parameter (maximum level number, or number of parallel passes).
r : A non-negative number of rounds.

Output:

D : A d-bit hash value $D = \mathcal{M}^{f}_{d,K,L,r}(M)$.

Procedure:

Initialize:

- Let $\ell = 0$, $M_0 = M$, and $m_0 = m$.

Main level-by-level loop:

- Let $\ell = \ell + 1$.
- If $\ell = L + 1$, return $\mathrm{SEQ}(M_{\ell-1}, d, K, L, r)$ as the hash function output.
- Let $M_\ell = \mathrm{PAR}(M_{\ell-1}, d, K, L, r, \ell)$. Let m_ℓ be the length of M_ℓ in bits.
- If $m_\ell = c$ (i.e. if M_ℓ is c words long), return the last d bits of M_ℓ as the hash function output. Otherwise, return to the top of the main level-by-level loop.

Figure 2-4: The MD6 Mode of Operation. With the default setting of $L = 31$, the SEQ operation is never used; the PAR operation is repeatedly called to reduce the input size by a factor of $b/c = 4$ until a single 16-word chunk remains.

The MD6 PAR Operation

Input:

$M_{\ell-1}$: A message of some non-negative length $m_{\ell-1}$ in bits.

d : The length d (in bits) of the desired hash output, $1 \leq d \leq 512$.

K : An arbitrary $k = 8$ word "key" value, containing a supplied key of *keylen* bytes.

L : A non-negative mode parameter (maximum level number, or number of parallel passes).

r : A non-negative number of rounds.

ℓ : A non-negative integer level number, $1 \leq \ell \leq L$.

Output:

M_ℓ : A message of length m_ℓ in bits, where $m_\ell = 1024 \cdot \max(1, \lceil m_{\ell-1}/4096 \rceil)$

Procedure:

Initialize:

- Let Q denote the array of length $q = 15$ words giving the fractional part of $\sqrt{6}$. (See [32, Appendix A].)
- Let f_r denote the MD6 compression function mapping a 64-word input data block B to a 16-word output chunk C using r rounds of computation. (f_r also takes 25 words of auxiliary input information.)

Shrink:

- Extend input $M_{\ell-1}$ if necessary (and only if necessary) by appending zero bits until its length becomes a positive integral multiple of $b = 64$ words. Then $M_{\ell-1}$ can be viewed as a sequence B_0, B_1, ..., B_{j-1} of b-word blocks, where $j = \max(1, \lceil m_{\ell-1}/bw \rceil)$.
- For each b-word block B_i, $i = 0, 1, \ldots, j-1$, compute C_i *in parallel* as follows:
  - Let p denote the number of padding bits in B_i; $0 \leq p \leq 4096$. (p can only be nonzero for $i = j - 1$.)
  - Let $z = 1$ if $j = 1$, otherwise let $z = 0$. ($z = 1$ only for the last block to be compressed in the complete MD6 computation.)
  - Let V be the one-word value $r \| L \| z \| p \| keylen \| d$ (see Figure 2-2).
  - Let $U = \ell \cdot 2^{56} + i$ be a "unique node ID"—a one-word value unique to this compression function operation.
  - Let $C_i = f_r(Q \| K \| U \| V \| B_i)$. ($C_i$ has length $c = 16$ words).
- Return $M_\ell = C_0 \| C_1 \| \ldots \| C_{j-1}$.

Figure 2-5: The MD6 PAR operator is a parallel compression operation producing level ℓ of the tree from level $\ell - 1$. With the default setting $L = 31$, this routine is used repeatedly to produce each higher layer of the tree, until the value at the root is produced.

The (Optional) MD6 SEQ Operation

Input:

M_L : A message of some non-negative length m_L in bits.
d : The length d (in bits) of the desired hash output, $1 \leq d \leq 512$.
K : An arbitrary $k = 8$ word key value, containing a supplied key of $keylen$ bytes.
L : A non-negative mode parameter (maximum tree height).
r : A non-negative number of rounds.

Output:

D : A d-bit hash value.

Procedure:

Initialize:

- Let Q denote the array of length $q = 15$ words giving the fractional part of $\sqrt{6}$. (See [32, Appendix A].)
- Let f_r denote the MD6 compression function mapping a 64-word data block B to a 16-word output block C using r rounds of computation. (f_r also takes 25 words of auxiliary input information.)

Main loop:

- Let C_{-1} be the zero vector of length $c = 16$ words. (This is the "IV".)
- Extend input M_L if necessary (and only if necessary) by appending zero bits until its length becomes a positive integral multiple of $(b - c) = 48$ words. Then M_L can be viewed as a sequence $B_0, B_1, \ldots, B_{j-1}$ of $(b - c)$-word blocks, where $j = \max(1, \lceil m_L/(b - c)w \rceil)$.
- For each $(b - c)$-word block B_i, $i = 0, 1, \ldots, j - 1$ *in sequence*, compute C_i as follows:
  - Let p be the number of padding bits in B_i; $0 \leq p \leq 3072$. (p can only be nonzero when $i = j - 1$.)
  - Let $z = 1$ if $i = j - 1$, otherwise let $z = 0$. ($z = 1$ only for the last block to be compressed in the complete MD6 computation.)
  - Let V be the one-word value $r \| L \| z \| p \| keylen \| d$ (see Figure 2-2).
  - Let $U = L \cdot 2^{56} + i$ be a "unique node ID"—a one-word value unique to this compression function operation.
  - Let $C_i = f_r(Q \| K \| U \| V \| C_{i-1} \| B_i)$. ($C_i$ has length $c = 16$ words).
- Return the last d bits of C_{j-1} as the hash function output.

Figure 2-6: The MD6 SEQ Operator is a sequential Merkle-Damgård-like hash operation producing a final hash output value. With the default setting of $L = 31$, SEQ is never used.

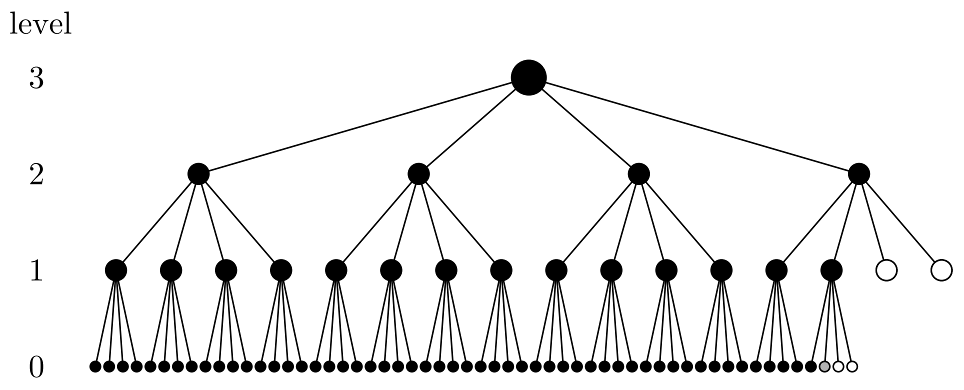

Figure 2-7: Structure of the standard MD6 mode of operation ($L = 31$). Computation proceeds from bottom to top: the input is on level 0, and the final hash value is output from the root of the tree. Each edge between two nodes represents a 16 word (128 byte, 1024-bit) chunk. Each small black dot on level 0 corresponds to a 16-word chunk of input message. The grey dot on level 0 corresponds to a partial chunk (less than 16 words) that is padded with zeros until it is 16 words long. A white dot (on any level) corresponds to a chunk of all zeros. Each medium or large black dot above level zero corresponds to an application of the compression function. The large black dot represents the final compression operation; here it is at the root. The final MD6 hash value is obtained by truncating the value computed there.

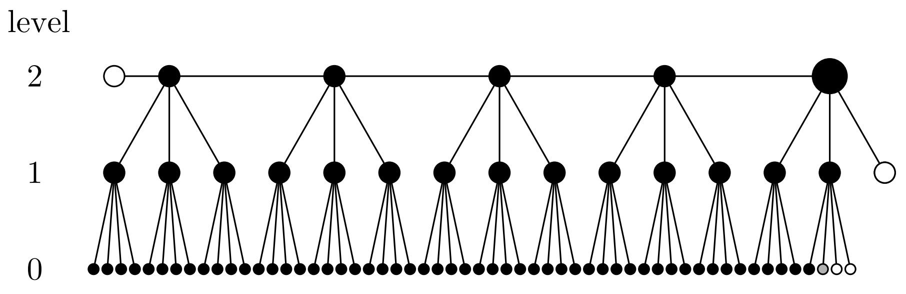

Figure 2-8: Structure of the MD6 mode of operation ($L = 1$). Computation proceeds from bottom to top and left to right; level 2 represents processing by SEQ. The hash function output is produced by the rightmost node on level 2. The white circle at the left on level 2 is the all-zero initialization vector for the sequential computation at that level.

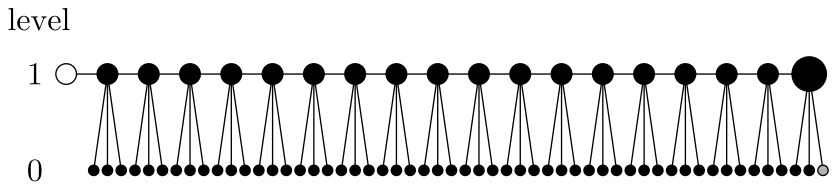

Figure 2-9: Structure of the MD6 mode of operation ($L = 0$). Computation proceeds from left to right only; level 1 represents processing by SEQ. The hash function output is produced by the rightmost node on level 1. This is similar to standard Merkle-Damgård processing. The white circle at the left on level 1 is the all-zero initialization vector for the sequential computation at that level.

In addition we will often use the following notational conventions when precisely defining the advantage of an adversary for attacking some cryptographic property of the compression function or hash function.

$$\mathbf{Adv}_{\mathsf{A}}^{P} = \Pr\left[\begin{array}{c} \text{Random values are chosen;} \\ \mathsf{A} \text{ is given these values as input;} \\ \mathsf{A} \text{ produces some output} \end{array} : \mathsf{A}\text{'s output violates } P\right]$$

For example, one definition of the advantage of A for breaking the collision resistance of f is due to Rogaway and Shrimpton [35].

$$\mathbf{Adv}_{\mathsf{A}}^{\text{fil-cr}} = \Pr\left[\begin{array}{c} K \overset{\$}{\leftarrow} \{0,1\}^k; \\ (m, m') \leftarrow \mathsf{A}(K) \end{array} : \begin{array}{c} m \neq m', \\ f(K, m) = f(K, m') \end{array}\right]$$

Here the probability is taken over the uniform random choice of key value K from the keyspace $\{0,1\}^k$ (where this random choice is denoted by \$). To be more precise, we should also take into account the internal randomness of the algorithm A, which in general is not a deterministic algorithm. We could denote this similarly as $(m, m') \overset{\$}{\leftarrow} \mathsf{A}(K)$, but throughout this paper we will often take this as a given and omit the \$.

2.2 Definitions

Definition 1 (Fixed Input Length)**.** A function f mapping domain $\mathcal{D}$ to $\mathcal{R}$ is a fixed input length (FIL) function if $\mathcal{D} = \{0,1\}^i$ for some positive integer i. That is, its inputs consist only of bit strings of some fixed length.

Definition 2 (Variable Input Length)**.** A function H mapping domain $\mathcal{D}$ to $\mathcal{R}$ is a variable input length (VIL) function if $\mathcal{D} = \{0,1\}^*$. That is, its inputs are bit strings of variable length.

Definition 3 (Domain Extender)**.** We say some algorithm A is a domain extender for property P if, given a fixed input length function f that is FIL-P (that is, has the property of P for fixed input length), then A^f (A when given oracle access to f) is a variable input length function that has the property of VIL-P (that is, it has the property of P for variable input length).

Note that traditionally a domain extender only extends the domain of the function and provides no guarantees that the new function satisfies any properties. However, in our context we will mean that A is a domain extender *and* a property P preserver.

Definition 4 (Running Time)**.** Oftentimes we will say that an algorithm A has "running time" t. By this we mean that t includes both the amount of time taken by an invocation of A as well as the size of the code implementing A, measured in some fixed RAM model of computation (as in [1]).

Definition 5 (Mode of Operation)**.** Throughout this paper we will use $\mathcal{M}$ to denote MD6's mode of operation. This is defined irrespective of the compression function used, although we will often use $\mathcal{M}^f : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^d$ to denote the MD6 mode of operation applied to the compression function f (i.e. $\mathcal{M}^f(M)$ is the MD6 hash of a message M), the superscript denoting that the mode of operation only makes black-box use of f. When needed, we will parameterize $\mathcal{M}^f$ by L, denoting this as $\mathcal{M}^f_L$.

On occasion we will consider the output of $\mathcal{M}^f$ without the final compression function. Recall that the final compression function has the z bit set to 1, and then its output is chopped from c bits down to d bits. Therefore we write the mode of operation without the final compression function as $\mathcal{H}^f : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$, where

$$\mathcal{M}^f = chop_d \circ f \circ \mathcal{H}^f.$$

The function $chop_d : \{0,1\}^c \to \{0,1\}^d$ operates simply by returning the *last* d-bits of its input. We can abbreviate the action of the final compression function and the chop by defining $g = (chop_d \circ f) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^d$.

Definition 6 (Height)**.** For a given maximum level parameter L, let the height of the MD6 hash tree parameterized by L on a message M of length $\mu = |M|$ be given as $height_L(\mu)$. Recall that the height of a tree is given by the distance of the root node from the farthest leaf node. For example, the tree in Figure 2-7 has height 3, the tree in Figure 2-8 has height 6, and the tree in Figure 2-9 has height 18.

Oftentimes we will have some adversary A that we grant q queries $M_1, M_2, \ldots, M_q$ to the MD6 hash algorithm $\mathcal{M}^f$, where we bound the total length of the queries, $\sum_i |M_i| \leq \mu$. One question we might ask is, in the course of execution of $\mathcal{M}^f$, how many queries to f will need to be made given the above resource constraints.

Lemma 2.1. Assume we are given oracle access to $f(K, \cdot)$ for some K. Then if we have q messages $M_1, M_2, \ldots, M_q$ such that

$$\sum_{i=1}^{q} |M_i| \leq \mu,$$

then in order to compute $D_i = \mathcal{M}_L^{f(K,\cdot)}(M_i)$ for all i we require at most $\delta(q, \mu)$ queries to the oracle $f(K, \cdot)$, where

$$\delta(q, \mu) = \frac{1}{3} \cdot \left\lceil \frac{\mu}{c} \right\rceil + q \log_4 \left(\frac{1}{q} \cdot \left\lceil \frac{\mu}{c} \right\rceil \right) + \frac{q}{3}.$$

Proof. First, note that in the worst case $L = 31$ and computing $\mathcal{M}^f$ consists of only PAR operations, so we will assume this case (and the bound will hold for smaller values of L). Now, for a given query M_i, let $\beta_i = \left\lceil \frac{|M_i|}{c} \right\rceil$ denote the number of c-word message blocks in M_i. If $\beta_i = 4^k$ for some integer k, then the hash tree computed by $\mathcal{M}^f$ is a complete height-k 4-ary tree. Therefore the number of queries to $f(K, \cdot)$ is the number of non-leaf nodes, which is at most $\frac{4^k-1}{3}$ (via geometric series), which satisfies the bound. If (in the worst case), $\beta_i = 4^k + 1$, then the hash tree computed by $\mathcal{M}^f$ consists of a root node with four sub-trees: a complete height-k 4-ary tree, a length-k path, and two leaf nodes. Thus the number of queries to $f(K, \cdot)$ is at most

$$\underbrace{\frac{4^k - 1}{3}}_{\text{height-}k\text{ subtree}} + k + 1 \leq \frac{\beta_i}{3} + \log_4 \beta_i + \frac{1}{3},$$

and therefore for queries $M_1, M_2, \ldots, M_q$,

$$\begin{aligned}\sum_{i=1}^{q} \left(\frac{\beta_i}{3} + \log_4 \beta_i + \frac{1}{3} \right) &= \frac{1}{3} \cdot \left\lceil \frac{\mu}{c} \right\rceil + \left(\sum_{i=1}^{q} \log_4 \beta_i \right) + \frac{q}{3} \\ &\leq \frac{1}{3} \cdot \left\lceil \frac{\mu}{c} \right\rceil + q \log_4 \left(\frac{1}{q} \cdot \left\lceil \frac{\mu}{c} \right\rceil \right) + \frac{q}{3}\end{aligned}$$

Where the last inequality follows due to the concavity of the log function. □

2.3 Related Work

The properties and applications of cryptographic hash functions have been well-studied in the literature. In particular, for an excellent overview of the history and uses of cryptographic hash functions the survey of Preneel [31] is a venerable, if possibly outdated, guide. A very thorough explanation and investigation of security properties for cryptographic hash functions is due to Rogaway and Shrimpton [35].

In the context of domain extension and property preservation, there are several important results in the literature; in particular, for variants of Merkle-Damgård [1, 14], CBC-MAC [6,7], and variants of CBC-MAC [16]. Tree-based modes of operation for hash functions have made several appearances in the literature; one such early work is due to Damgård [15]. Recently, Sarkar and Schellenberg proposed a tree-based hash algorithm, although theirs differs from MD6 in many respects [36]. To date, this is the only known large-scale work we are aware of regarding the domain extension and property preserving characteristics of a tree-based hash.

Chapter 3

Collision and Preimage Resistance

In this chapter we prove certain results about the collision and preimage resistance of the MD6 hash function mode of operation. In particular, we show that the MD6 mode of operation acts as a *domain extender* for various properties of the fixed input length compression function used. Our goal is to show that if we assume that the compression function is collision resistant (respectively, preimage resistant), then the entire hash function will be collision resistant (respectively, preimage resistant) as well.

One important caveat is that the notions of collision resistance and preimage resistance are only defined for keyed hash functions. As observed by Rogaway [34], an unkeyed hash function $H : \{0,1\}^* \to \{0,1\}^d$ is trivially non-collision resistant: the existence of an efficient algorithm that can find distinct strings M and M' such that $H(M) = H(M')$ is guaranteed, simply by virtue of the existence of such collisions (as Rogaway says, the algorithm has the collision "hardwired in"). Therefore for an unkeyed hash function H, what is meant by saying that H is collision resistant is not that an efficient collision-finding algorithm does not exist, but rather that no efficient collision-finding algorithm exists that is *known to man* (the emphasis is Rogaway's). Similar concerns can also be expressed for the preimage resistance of unkeyed hash functions. Although MD6 can behave as a keyed hash function, certain applications call for the use of the unkeyed variant (where the key field is simply set to λ, the empty string), such as any application that uses a public hash function. Thus we would like to argue, with some amount of rigor, that its unkeyed variant is collision or preimage resistant as well.

Fortunately, we can perform reductions for finding collisions and preimages. Specifically, we will show that if one has a collision or preimage for the entire hash, then one can construct a collision or preimage for the underlying compression function. Therefore if one assumes that there is no known algorithm for finding collisions or preimages in the compression function, then there should be no known algorithm for finding collisions or preimages in the overall hash function. While this is not a completely rigorous notion of security (as it relies on the extent of human knowledge for finding collisions, which is impossible to formalize), it is the best we can do in these circumstances.

After proving reductions for these properties, we show that they apply to the keyed hash function variant of MD6 as well (for certain — now rigorous — definitions of collision resistance and preimage resistance). That is, we will demonstrate that if an algorithm exists that can break the property of collision resistance or preimage resistance for keyed MD6, then we can use this algorithm as a black box for breaking the collision resistance or preimage resistance of the underlying compression function. We may then conclude that breaking either property is at least as difficult as breaking the respective property for the compression function.

Some of the proofs in this section are similar to those of Sarkar and Schellenberg [36], owing to the fact that their mode of operation is also based on a tree-like construction. However, the differences between the MD6 mode of operation and that of Sarkar and Schellenberg are significant enough to warrant entirely new proofs of security.

3.1 Collision Resistance

To begin, we first define what it means for a keyed fixed input length (FIL) compression function f to be collision resistant, and what it means for a keyed variable input length (VIL) hash function H to be collision resistant.

Definition 7 (FIL-Collision Resistance)**.** For a keyed FIL function $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$, define the advantage of an adversary A for finding a collision as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{fil-cr}} = \Pr\left[\begin{array}{ccc} K \xleftarrow{\$} \{0,1\}^k; & \textbf{:} & m \neq m', \\ (m,m') \leftarrow \mathsf{A}(K) & & f(K,m) = f(K,m') \end{array}\right]$$

We define the insecurity of f with respect to FIL-collision resistance (FIL-CR) as

$$\mathbf{InSec}_f^{\text{fil-cr}}(t) = \max_{\mathsf{A}} \left\{\mathbf{Adv}_{\mathsf{A}}^{\text{fil-cr}}\right\},$$

when the maximum is taken over all adversaries A with total running time t.

Definition 8 (VIL-Collision Resistance)**.** For a keyed VIL function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^c$, define the advantage of an adversary A for finding a collision as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{vil-cr}} = \Pr\left[\begin{array}{ccc} K \xleftarrow{\$} \{0,1\}^k; & \textbf{:} & M \neq M', \\ (M,M') \leftarrow \mathsf{A}(K) & & H(K,M) = H(K,M') \end{array}\right]$$

We define the insecurity of f with respect to VIL-collision resistance (VIL-CR) as

$$\mathbf{InSec}_H^{\text{vil-cr}}(t) = \max_{\mathsf{A}} \left\{\mathbf{Adv}_{\mathsf{A}}^{\text{vil-cr}}\right\},$$

when the maximum is taken over all adversaries A with total running time t.

Here we add the additional assumption that A must have computed the hashes of these messages $H(K,M)$ and $H(K,M')$ at some point (for the message pair (M,M') that it returned), and so the total time allotment t includes the cost of computing these hashes. This is a reasonable assumption — not without precedent [1, Section 4.1] — as A should at least verify that $H(K,M) = H(K,M')$.

We now give a series lemmas that culminate in demonstrating how one can use a collision in MD6 to construct a collision in the underlying compression function. These lemmas rely heavily on the detailed description of the MD6 mode of operation given in Section 2.1.1. We give a sketch of our approach as follows. The MD6 mode of operation makes use of two operations, PAR and SEQ. The global parameter L determines the maximum number of times that the operation PAR is invoked (recursively) on the input message. If the length of the input is long enough that after (up to) L iterations of PAR the resulting output is still larger than d bits, the SEQ operation is performed to bring the final digest to d bits. Therefore our lemmas will be of the following form: if we have two messages that form a collision in PAR or SEQ, we can examine the intermediary values produced in the computation of PAR or SEQ to find a collision either in the compression function f or in the final compression function $g = chop_d \circ f$. We will conclude by noticing that if there is a collision in the overall MD6 hash, then there must be a collision in one of the PAR or SEQ operations.

Lemma 3.1. Suppose $M, M' \in \{0,1\}^*$ are distinct and further, $\mathrm{SEQ}(M, d, K, L, r) = \mathrm{SEQ}(M', d, K, L, r)$. Then we can construct $m, m' \in \{0,1\}^n$ with $m \neq m'$ such that $f(K, m) = f(K, m')$ or $g(K, m) = g(K, m')$.

Proof. Recall from Section 2.1.1 that the MD6 SEQ operation performs analogously to the Merkle-Damgård construction, and therefore this proof will proceed similarly to the well-established collision resistance proofs of security [15]. As shown in Figure 2-6, we begin by padding out M and M' with zeroes to be a positive integral multiple of $(b - c) = 48$ words, and subdivide each message into sequences $M = B_0 \| B_1 \| \cdots \| B_{j-1}$ and $M' = B'_0 \| B'_1 \| \cdots \| B'_{j'-1}$ of j and j' (respectively) $(b-c)$-word blocks. Before we can proceed, we must first define some intermediary values that we use throughout our proof. Let

$$m_i = U_i \| V_i \| C_{i-1} \| B_i,$$

where we define the auxiliary fields,

$$\begin{aligned}
U_i &= L \cdot 2^{56} + i \\
V_i &= r \| L \| z_i \| p_i \| keylen \| d \\
z_i &= \begin{cases} 1, & \text{if } i = j-1 \\ 0, & \text{otherwise} \end{cases} \\
p_i &= \begin{cases} \#\text{ of padding bits in } B_{j-1}, & \text{if } i = j-1 \\ 0, & \text{if } i < j-1 \end{cases}
\end{aligned}$$

and the chaining variables,

$$\begin{aligned}
C_{-1} &= 0^{64w} \\
C_i &= f(K, m_i), \text{ for } 0 \leq i < j-1 \\
C_{j-1} &= g(K, m_{j-1}) = chop_d(f(K, m_{j-1})).
\end{aligned}$$

We also define m'_i similarly for the B'_i values.

Now first suppose that $j \neq j'$; that is, the number of blocks in the input messages differ. Then we are easily able to construct a collision between the inputs to the final compression function in SEQ. Since $j \neq j'$ we have $U_{j-1} \neq U'_{j'-1}$, as the binary representation of $j-1$ and $j'-1$ are encoded into these values. This clearly implies that $m_{j-1} \neq m'_{j'-1}$. Moreover, we have our collision, since

$$g(K, m_{j-1}) = C_{j-1} = \text{SEQ}(M, d, K, L, r) = \text{SEQ}(M', d, K, L, r) = C'_{j'-1} = g(K, m'_{j'-1}).$$

On the other hand, suppose that $j = j'$ and the number of blocks in the input messages are the same. We show that at some point along the chain of computations, a collision occurs. That is, the event $C_i = C'_i$ implies that either there is a collision at block i, or the previous chaining variables are equal, $C_{i-1} = C'_{i-1}$. Thus starting from the assumption that $C_{j-1} = C'_{j-1}$ (i.e. that the outputs of SEQ are equal), we can work our way backwards through the chain to find a collision.

First, suppose that $m_{j-1} \neq m_{j-1}$. Since $g(K, m_{j-1}) = C_{j-1} = C'_{j-1} = g(K, m'_{j-1})$, we have a collision. Now, on the contrary, suppose that $m_{j-1} = m'_{j-1}$. Then this implies that $C_{j-2} = C'_{j-2}$.

Now, fix some i such that $0 \leq i < j - 1$ and suppose that $C_i = C'_i$. Then either $m_i \neq m'_i$ and we have a collision $f(K, m_i) = C_i = C'_i = f(K, m'_i)$, or $m_i = m'_i$ and therefore $C_{i-1} = C'_{i-1}$. Thus by starting from $C_{j-1} = C'_{j-1}$ and walking backwards along the chain of computations, we either find a collision or, for each i, $m_i = m'_i$. In particular, if $m_i = m'_i$ for all i, then this implies that $B_i = B'_i$ for all i as well. Furthermore, this implies that the number of padding bits are equal, $p_{j-1} = p'_{j-1}$. However, if this is the case then it must be that $M = M'$, which is a contradiction. Therefore a collision in f or g must occur at some point along the chain. $\square$

Lemma 3.2. Suppose $M, M' \in \{0,1\}^*$ are distinct and further, $\text{PAR}(M, d, K, L, r, \ell) = \text{PAR}(M', d, K, L, r, \ell)$. Then we can construct $m, m' \in \{0,1\}^n$ with $m \neq m'$ such that $f(K, m) = f(K, m')$.

Proof. We proceed much in the same fashion as Lemma 3.1, but this proof is simpler because of the parallel nature of PAR. Following the definition of PAR shown in Figure 2-5, we pad out M and M' with zeroes until their lengths are multiples of $b = 64$ words, and subdivide each message into sequences $M = B_0 \| B_1 \| \cdots \| B_{j-1}$ and $M' = B'_0 \| B'_1 \| \cdots \| B'_{j'-1}$ of j and j' (respectively) b-word blocks. As before, we define the intermediary variables we will use in this proof. Let

$$m_i = U_i \| V_i \| B_i,$$

where we define the auxiliary fields,

$$U_i = \ell \cdot 2^{56} + i$$

$$V_i = r \| L \| z \| p_i \| keylen \| d$$

$$z = \begin{cases} 1, & \text{if } j = 1 \\ 0, & \text{otherwise} \end{cases}$$

$$p_i = \begin{cases} \# \text{ of padding bits in } B_{j-1}, & \text{if } i = j-1 \\ 0, & \text{if } i < j-1 \end{cases}$$

and the output variables,

$$C_i = f(K, m_i)$$

We also define m'_i similarly for the B'_i values.

From our definitions,

$$C_0 \| C_1 \| \cdots \| C_{j-1} = \text{PAR}(M, d, K, L, r, \ell) = \text{PAR}(M', d, K, L, r, \ell) = C'_0 \| C'_1 \| \cdots \| C'_{j'-1},$$

and therefore $j = j'$.

Moreover, if there exists an i such that $m_i \neq m'_i$, then we have a collision, as $f(K, m_i) = C_i = C'_i = f(K, m'_i)$. Now suppose that for all i, $m_i = m'_i$. This implies that for all i, the input messages $B_i = B'_i$ and the number of padding bits $p_i = p'_i$. Therefore it must be the case that $M = M'$, which is a contradiction. □

Theorem 3.3. Suppose $M, M' \in \{0,1\}^*$ and L, L', K, K' provide a collision in the hash function $\mathcal{M}^f$; that is, $\mathcal{M}^f_L(K, M) = \mathcal{M}^f_{L'}(K', M')$. Then we can construct $m, m' \in \{0,1\}^n$ with $m \neq m'$ such that $f(K, m) = f(K', m')$ or $g(K, m) = g(K', m')$.

PROOF. Let ℓ and ℓ' be the "layered height" of each hash tree. That is, in the computation of MD6, ℓ and ℓ' are the number of total applications of PAR and SEQ on m and m', respectively. Note this is not the height of the hash tree, since the root node of the graph in Figure 2-8 has height 6. Rather, it is the "level" of the root node in the computation, plus one if SEQ has been applied. Thus the layered height of the hash tree in Figure 2-8 is 3, as is the layered height of the hash tree in Figure 2-7. From this point, we assume that $L = L'$, $K = K'$, and $\ell = \ell'$, since these parameters (L, K, ℓ) are included as part of the input to the final compression function g. If this is not the case, then there is a collision in g and we are done. Thus we drop the prime $'$ in the variable names and consider only L, K, and ℓ.

The rest of the proof will follow substantially from Lemmas 3.1 and 3.2. As in the definition of MD6 (see Figure 2-4), we use the following intermediary variables,

$$
\begin{aligned}
M_0 &= M \\
M_i &= \mathrm{PAR}(M_{i-1}, d, K, L, r, i), \text{ for } 1 \leq i < \ell \\
D = M_\ell &= \begin{cases} \mathrm{SEQ}(M_{\ell-1}, d, K, L, r), & \text{if } \ell = L + 1 \\ \mathit{chop}_d(\mathrm{PAR}(M_{\ell-1}, d, K, L, r, i)), & \text{otherwise} \end{cases}
\end{aligned}
$$

with D' and M'_i defined similarly for M'.

By Lemma 3.2, $M_i \neq M'_i$ implies that either $M_{i+1} \neq M'_{i+1}$ or we can find a collision in one of the compression functions f used at level i. Therefore, moving from the bottom of the hash tree up (starting from the condition $M_0 \neq M'_0$) we either find a collision in f or reach level $\ell - 1$ with $M_{\ell-1} \neq M'_{\ell-1}$. If PAR is the last function executed (i.e. $\ell < L + 1$), then by Lemma 3.2 we have found a collision in g, since $D = D'$. If SEQ is the last function executed (i.e. $\ell = L + 1$), then by Lemma 3.1 we have also found a collision in either f or g, again because $D = D'$. □

Theorem 3.4. Let $f : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^c$ be a FIL-CR function, and suppose that $g = (chop_d \circ f) : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^d$ is also FIL-CR. Then $\mathcal{M}^f : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^d$ is a VIL-CR function with

$$\mathbf{InSec}^{\text{vil-cr}}_{\mathcal{M}^f}(t) \leq \mathbf{InSec}^{\text{fil-cr}}_{f}(2t) + \mathbf{InSec}^{\text{fil-cr}}_{g}(2t).$$

Proof. Suppose A is an algorithm with the best possible chance of success for breaking the VIL-collision resistance of $\mathcal{M}^f$ among all algorithms running in time t, so that the probability of success of A is $\mathbf{InSec}^{\text{vil-cr}}_{\mathcal{M}^f}(t)$. We construct an algorithm C that uses A as a subroutine to attack the FIL-collision resistance of either f or g.

To begin, C receives as input the hash function key K; in order to succeed, it must produce messages $m \neq m'$ such that $f(K,m) = f(K,m')$ or $g(K,m) = g(K,m')$, and run in time no greater than $2t$. C then invokes A on the key K, which ultimately returns messages $M \neq M'$ such that $\mathcal{M}^f(K,M) = \mathcal{M}^f(K,M')$.

Algorithm C
Input: K, the compression function key; A, the collision-finding algorithm
Output: $m \neq m'$, such that $f(K,m) = f(K,m')$ or $g(K,m) = g(K,m')$

```
1.  $(M, M') \leftarrow \mathsf{A}(K)$
2.  $D \leftarrow \mathcal{M}^f(K,M)$ and store each input to $f(K,\cdot)$
3.  $D' \leftarrow \mathcal{M}^f(K,M')$ and store each input to $f(K,\cdot)$
4.  if $M \neq M'$ and $D = D'$
5.      then Use Theorem 3.3 to find $m \neq m'$ that collide in $f(K,\cdot)$ or $g(K,\cdot)$
6.          return $(m, m')$
7.      else return $(0^n, 0^n)$
```

By Theorem 3.3, from a collision in $\mathcal{M}^f(K,\cdot)$ we can recover a collision in $f(K,\cdot)$ or $g(K,\cdot)$. This recovery process takes time at most t, since it only requires computing the hashes $\mathcal{M}^f(K,M)$ and $\mathcal{M}^f(K,M')$ and considering the intermediate values queried to $f(K,\cdot)$ in each computation. In addition, the running time of the algorithm A is at most t. Thus with probability of success at least $\mathbf{InSec}^{\text{vil-cr}}_{\mathcal{M}^f}(t)$, we are able to find a collision in either f or g in time $2t$. □

3.2 First-Preimage Resistance

Recall that the property of first-preimage resistance is an important one in many cryptographic applications. For example, as mentioned in Chapter 1, most computer systems store the hashes of user passwords. The inability of any adversary to invert these hashes is therefore important to preserve the security of the system[1].

Since we aim to show that the MD6 mode of operation extends the property of first-preimage resistance from the compression function to the overall hash function, we precisely define what it means for both the FIL compression function and the VIL hash function to be first-preimage resistant.

Definition 9 (FIL-Preimage Resistance)**.** For a keyed FIL function $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$, define the advantage of an adversary A for finding a preimage as

$$\begin{aligned}\mathbf{Adv}_{\mathsf{A}}^{\text{fil-pr}}(D) &= \Pr\Big[K \xleftarrow{\$} \{0,1\}^k; m \leftarrow \mathsf{A}(K,D) : \quad f(K,m) = D\Big],\\ \mathbf{Adv}_{\mathsf{A}}^{\text{fil-pr}} &= \max_{D\in\{0,1\}^d}\Big\{\mathbf{Adv}_{\mathsf{A}}^{\text{fil-pr}}(D)\Big\}.\end{aligned}$$

We define the insecurity of f with respect to FIL-preimage resistance (FIL-PR) as

$$\mathbf{InSec}_f^{\text{fil-pr}}(t) = \max_{\mathsf{A}}\Big\{\mathbf{Adv}_{\mathsf{A}}^{\text{fil-pr}}\Big\},$$

when the maximum is taken over all adversaries A with total running time t.

Definition 10 (VIL-Preimage Resistance)**.** For a keyed VIL function $H : \{0,1\}^k \times$

[1]Although in practice, many systems use a salted hash for their password file to make dictionary-based inversion attacks much more difficult.

$\{0,1\}^* \to \{0,1\}^d$, define the advantage of an adversary A for finding a preimage as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{vil-pr}}(D) = \Pr\Big[K \xleftarrow{\$} \{0,1\}^k; M \leftarrow \mathsf{A}(K,D) : \quad H(K,M) = D\Big],$$
$$\mathbf{Adv}_{\mathsf{A}}^{\text{vil-pr}} = \max_{D\in\{0,1\}^d} \Big\{\mathbf{Adv}_{\mathsf{A}}^{\text{vil-pr}}(D)\Big\}.$$

We define the insecurity of H with respect to VIL-preimage resistance (VIL-PR) as

$$\mathbf{InSec}_{H}^{\text{vil-pr}}(t) = \max_{\mathsf{A}} \Big\{\mathbf{Adv}_{\mathsf{A}}^{\text{vil-pr}}\Big\},$$

when the maximum is taken over all adversaries A with total running time t. As before, we make the reasonable assumption that computing $H(K,M)$ counts towards the total time allotment of t, since we assume any preimage-finding algorithm must at least verify that M is a valid preimage of D.

Note that these definitions differ from the commonly regarded notion of preimage resistance (Pre) as defined by Rogaway and Shrimpton [35]:

$$\mathbf{Adv}_{\mathsf{A}}^{\text{Pre}} = \Pr\left[\begin{array}{c} K \xleftarrow{\$} \{0,1\}^k; M \xleftarrow{\$} \{0,1\}^n; \\ D \leftarrow f(K,M); M' \leftarrow \mathsf{A}(K,D) \end{array} : \quad f(K,M') = D\right]$$

In particular, the definitions given above are a stricter notion of preimage resistance that Rogaway and Shrimpton term "everywhere preimage-resistance" (ePre); this definition attempts to capture the infeasibility of finding a preimage for a given D, over *all* choices of D. We adopt this definition because it simplifies our analysis. Since everywhere preimage-resistance implies preimage resistance [35], we do not lose any security by doing so.

We begin by demonstrating, via reduction, that the MD6 hash function is VIL-preimage resistant so long as its underlying compression function is FIL-preimage resistant.

Theorem 3.5. Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$, and suppose that $g = (chop_d \circ f) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^d$ is also FIL-PR. Then $\mathcal{M}^f : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^d$ is a VIL-PR function with

$$\mathbf{InSec}^{\text{vil-pr}}_{\mathcal{M}^f}(t) \leq \mathbf{InSec}^{\text{fil-pr}}_{g}(2t).$$

Proof. Suppose that A is an algorithm with the best possible chance of success for breaking the VIL-preimage resistance of $\mathcal{M}^f$ among all algorithms running in time t, so that the probability of success of A is $\mathbf{InSec}^{\text{vil-cr}}_{\mathcal{M}^f}(t)$. We construct a new algorithm P, running in time at most $2t$, that uses A as a subroutine to attack the FIL-preimage resistance of g.

The behavior of P is straightforward: if A manages to find a valid preimage, then P can simply compute the hash and return the input to the final compression function g. However, some care must be taken in the analysis because, as mentioned in the definition of preimage resistance, the choice of target digest D that maximizes the advantage *can* depend on the algorithm P used.

```
Algorithm P
Input: $K$, the compression function key; $\mathsf{A}$, the preimage-finding algorithm; $D$, the
    target digest
Output: $m$, such that $g(K, m) = D$
1.  $M \leftarrow \mathsf{A}(K, D)$
2.  $D' \leftarrow \mathcal{M}^f(K, M)$
3.  $m \leftarrow$ the input to the final compression $g(K, \cdot)$ in the computation of $D'$
4.  if $D = D'$
5.      then return $m$
6.      else return $0^n$
```

To begin with, P receives as input a digest $D \in \{0,1\}^d$, the key K, and the $\mathcal{M}^f$ preimage-finding algorithm A, and its goal is to produce a preimage $m \in \{0,1\}^n$ such that $g(K, m) = D$. Next, P invokes A on the target digest D and key K to receive a message $M \in \{0,1\}^*$ such that, with some probability, $\mathcal{M}^f(K, M) = D$. If M is indeed a preimage of D, then letting $m \in \{0,1\}^n$ be the input given to the final compression function $g(K,\cdot)$ in the computation of $\mathcal{M}^f(K,M)$ will indeed give a preimage of D.

Now we wish to show that the advantage of P is at least the advantage of A. Let $\widehat{D}$ be the value of the target digest D that maximizes the advantage of A:

$$\widehat{D} = \underset{D\in\{0,1\}^d}{\arg\max}\left\{\Pr\left[K \xleftarrow{\$} \{0,1\}^k; M \leftarrow \mathsf{A}(K,D): \quad \mathcal{M}^f(K,M) = D\right]\right. .$$

Then the advantage of P when given target digest $\widehat{D}$ is at most its advantage over the best possible D, so that

$$\mathbf{InSec}^{\text{vil-pr}}_{\mathcal{M}^f}(t) \leq \mathbf{Adv}^{\text{vil-pr}}_{\mathsf{A}} = \mathbf{Adv}^{\text{vil-pr}}_{\mathsf{A}}(\widehat{D}) \leq \mathbf{Adv}^{\text{fil-pr}}_{\mathsf{P}}(\widehat{D}) \leq \mathbf{Adv}^{\text{fil-pr}}_{\mathsf{P}} \leq \mathbf{InSec}^{\text{fil-pr}}_{g}(2t).$$

In order to prove the bound for running time, notice that it takes time at most t to run $\mathsf{A}(K,D)$, and by our earlier assumption it takes time at most t to compute $\mathcal{M}^f(K,M)$. Therefore the total time is at most $2t$.

$\square$

3.3 Second-Preimage Resistance

As discussed in Chapter 1, second-preimage resistance is defined as the computational infeasibility of any adversary, given a target message m, to produce a different message m' such that these two messages hash to the same value. Clearly, second-preimage resistance is a potentially *stronger* assumption than collision resistance, since producing second preimages also yields hash function collisions. Therefore, as in other treatments of this problem [36], it suffices in general to prove collision resistance, which we demonstrated earlier in Section 3.1.

Unfortunately, trying to prove a reduction of the FIL-second-preimage resistance of the compression function to the VIL-second-preimage resistance of the overall hash function fails to work naturally. The problem with the reduction is that we have some algorithm A that can break the second-preimage resistance of $\mathcal{M}^f$ and we want to construct an algorithm S that uses A to break the second-preimage resistance of f. So S receives a message $m \in \{0,1\}^n$ and a key $K \in \{0,1\}^k$ and must find $m' \in \{0,1\}^n$ such that $m \neq m'$ and $f(K,m) = f(K,m')$. However, attempting to invoke A on m directly will not succeed. In particular, A is only guaranteed to succeed with some probability over the choice of K and $M \in \{0,1\}^*$. In particular, A could be excellent at finding second-preimages when given a target M such that $|M| > n$, but absolutely miserable when $|M| = n$. Therefore we are unable to translate the success of A into the success of S and the reduction fails.

Although it seems like it should be possible to perform such a reduction, we know of no approach for reducing the property successfully. In addition, we do not know of any similar attempts in the literature to prove domain extension for second-preimage resistance. Therefore we will simply say that the collision resistance of MD6 is secure with $d/2$ bits of security, therefore it follows that the second-preimage resistance of MD6 is secure with at least $d/2$ bits of security and hope that suffices.

However, we would like to specifically address the security requirements in the NIST SHA-3 hash function specifications [29], with respect to second-preimage resistance. Recall from Section 1.2 that for a hash function with a d bit message digest, then for a target preimage of 2^k bits the hash function should have second-preimage security of approximately $d - k$ bits.

Recall that the rationale for this condition is the following. In a hash function with an iterative mode of operation (such as the plain Merkle-Damgård construction), a target preimage m consisting of 2^k message blocks forms a chain of 2^k invocations of the compression function. Therefore an adversary wishing to perform a second preimage attack on m can simply pick a random r and compute $f(IV, r) = y$. It can then check whether y matches any of the 2^k compression function outputs y_i. If so, it outputs the message $m' = r \| m_{i+1} \| \cdots \| m_{2^k}$, which is a valid collision with m.

This attack works only because the length of the message m is not encoded in the hash, so an attacker is able to substitute the prefix r anywhere into the message chain. One simple method to foil this attack is to append the length of the message to the end of the message, which prevents an attacker from being able to substitute a truncated message that collides with m. However, even this approach succumbs to similar cryptographic attacks, as demonstrated by Kelsey and Schneier [20].

MD6 behaves differently from these approaches. Each compression function used is given control words U and V in the input that label each compression function with its position in the hash tree. Therefore the above attacks against Merkle-Damgård and strengthened Merkle-Damgård are foiled since the adversary is no longer given the 2^k-for-1 advantage that it enjoyed for the Merkle-Damgård mode of operation (effectively, it is not able to query a substructure of the hash function).

Chapter 4

Pseudorandomness

Pseudorandomness is a useful property for a hash function to have. For one, pseudorandomness implies unpredictability, meaning a pseudorandom hash function can perform as a message authentication code (MAC) [18,19]. In addition, many cryptographic protocols are proved to be secure in the so-called "random oracle model" [8, 17], which assumes the existence of an oracle that maps $\{0,1\}^*$ into some fixed output domain $\mathcal{D}$. The oracle is a black box that responds to queries in $\{0,1\}^*$ with a uniformly random response chosen from $\mathcal{D}$ (except for inputs that have been queried previously, whereupon it is consistent). In practice, protocols that assume the existence of a random oracle use a cryptographic hash function instead, in which case we desire that the hash function family be pseudorandom or in some sense indistinguishable from a random oracle. Unfortunately, random oracles do not actually exist in real life, and therefore proofs in the random oracle model only provide a heuristic for security [12]. Nevertheless, it is still desirable to be able to show that a cryptographic hash function family is pseudorandom, under certain assumptions on the compression function used.

Previous works have shown that the Cipher Block Chaining mode of operation is a domain extender for the property of pseudorandomness [6,7,21]. In this chapter we will demonstrate that the MD6 mode of operation also acts as a domain extender for fixed-length pseudorandom functions.

4.1 Maurer's Random System Framework

Throughout much of this section we use key concepts from the Random Systems framework developed by Ueli Maurer [21]. Many of these definitions and lemmas and much of the terminology are composites of several related papers [21, 22, 30]

4.1.1 Notation

We generally adhere to the notation used in previous work. Characters in a calligraphic font (such as $\mathcal{X}$ or $\mathcal{Y}$) denote sets, and their corresponding italicized roman characters X and Y denote random variables that take values in $\mathcal{X}$ and $\mathcal{Y}$ (with some distribution). Superscripts for sets and random variables generally denote a tuple, so $\mathcal{X}^i = \underbrace{\mathcal{X} \times \cdots \times \mathcal{X}}_{i}$ and $X^i = (X_1, \ldots, X_i)$ is a random variable over $\mathcal{X}^i$. We reserve bold-face characters for random systems, which are defined below.

4.1.2 Definitions

In order to reason about the complicated interactions of certain cryptographic systems, it is helpful to use the Random Systems framework of Maurer [21]. In particular, for some cryptographic system $\mathbf{S}$ and for each i, $\mathbf{S}$ takes in an input X_i and produces (probabilistically) a corresponding output Y_i (in sequence, so next it takes input X_{i+1} and produces Y_{i+1}). If $\mathbf{S}$ is stateless, then Y_i depends only on X_i; however, we can also consider $\mathbf{S}$ as possibly stateful, and so Y_i depends on the totality of the previous values $X_1, X_2, \ldots, X_i$ and $Y_1, Y_2, \ldots, Y_{i-1}$ (which are referred to as X^i and Y^{i-1} for convenience).

Thus the behavior of the random system $\mathbf{S}$ can be defined as an sequence of conditional probability distributions, as follows.

Definition 11 (Random System)**.** A $(\mathcal{X}, \mathcal{Y})$-random system $\mathbf{F}$ is an infinite sequence of conditional probability distributions

$$\mathbf{F} = \left\{ \mathsf{Pr}_{Y_i | X^i Y^{i-1}} \right\}_{i=1}^{\infty}.$$

Collectively, we denote this sequence as $\mathsf{Pr}^{\mathbf{F}}_{Y_i|X^iY^{i-1}}$, the superscript denoting which random system this distribution corresponds to.

Definition 12 (Equivalence of Random Systems)**.** Two random systems $\mathbf{F}$ and $\mathbf{G}$ are said to be equivalent, written $\mathbf{F} \equiv \mathbf{G}$, if

$$\text{for all } i \geq 1,\ \mathsf{Pr}^{\mathbf{F}}_{Y_i|X^iY^{i-1}} \equiv \mathsf{Pr}^{\mathbf{G}}_{Y_i|X^iY^{i-1}},$$

or equivalently, for all $i \geq 1, (y_1, \ldots, y_i) \in \mathcal{Y}^i, (x_1, \ldots, x_i) \in \mathcal{X}^i$,

$$\mathsf{Pr}^{\mathbf{F}}_{Y_i|X^iY^{i-1}}(x_1, \ldots, x_i, y_1, \ldots, y_i) = \mathsf{Pr}^{\mathbf{G}}_{Y_i|X^iY^{i-1}}(x_1, \ldots, x_i, y_1, \ldots, y_i).$$

Definition 13 (Random Function)**.** A random function $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ is a random variable that takes as values functions on $\mathcal{X} \to \mathcal{Y}$ (with some given distribution). Therefore $\mathbf{F}$ is also a (stateless) random system, where

$$\mathsf{Pr}^{\mathbf{F}}_{Y_i|X^iY^{i-1}} = \mathsf{Pr}^{\mathbf{F}}_{Y_i|X_i}$$

and this distribution is determined by the distribution of $\mathbf{F}$ on $\mathcal{X} \to \mathcal{Y}$.

Example 1. Consider the following random functions $\mathbf{R}$ and $\mathbf{P}$.

Uniform Random Function Let $\mathbf{R} : \mathcal{X} \to \mathcal{Y}$ denote the random function with a uniform distribution over the space of all functions mapping $\mathcal{X} \to \mathcal{Y}$.

Uniform Random Permutation Let $\mathbf{P} : \mathcal{X} \to \mathcal{X}$ denote the random function with a uniform distribution over the space of all permutations mapping $\mathcal{X} \to \mathcal{X}$.

It is a well-known fact that if we are only given $o\left(\sqrt{|\mathcal{X}|}\right)$ queries, it is difficult to distinguish between a uniform random function $\mathbf{R} : \mathcal{X} \to \mathcal{X}$ and a uniform random permutation $\mathbf{P} : \mathcal{X} \to \mathcal{X}$. We will prove this fact via the random system framework in Example 2.

We now attempt to develop the formal notion of *monotone conditions* for random systems. Intuitively, we are trying to capture some series of events that occur on the choices of inputs and outputs of the random system. For example, suppose we are trying to distinguish between $\mathbf{R}$ and $\mathbf{P}$ given q queries, as above. If we condition on the event that we have not observed any collisions in the output of $\mathbf{R}$, then the distribution on the outputs of each random function are identical (and therefore we have no hope of being able to distinguish them). Thus in this example we might say that the monotone condition is "the event that we have not observed a collision in the output of $\mathbf{R}$ up to query i". We can formalize this intuitive notion as follows.

Definition 14 (Monotone Conditions)**.** A monotone condition $\mathcal{A}$ for a random system $\mathbf{F}$ is an infinite sequence $(A_1, A_2, \dots)$ of events with an additional monotonicity condition. We define A_i to be the event that the specified condition is satisfied after query i, and $\overline{A_i}$ is the negation of this event (the specified condition is not satisfied after query i). The monotonicity of the condition $\mathcal{A}$ means that once the event is not satisfied for a given query i, it will not be satisfied after further queries (so, $\overline{A_i} \implies \overline{A_{i+1}}$).

We can additionally define the random system $\mathbf{F}$ conditioned on $\mathcal{A}$, $\mathbf{F} \,|\, \mathcal{A}$, to be the sequence of conditional probability distributions

$$\mathsf{Pr}^{\mathbf{F}}_{Y_i|X^iY^{i-1}A_i}, \text{ for all } i \geq 1$$

which are simply the distribution on the output Y_i conditioned on the previous state $X^{i-1}Y^{i-1}$, the current query X_i, and the monotone condition A_i. Note that we do not need to condition on the event $A_1 \wedge \cdots \wedge A_i$, since $A_i \implies A_{i-1}$. For a more formal definition, see [21, 22].

To go back to our earlier example, it is clear that the no-collisions condition is monotone, because observing a collision after query i implies that a collision has been observed after any further queries. As we will show in Example 2, when we condition $\mathbf{R}$ on this no-collision monotone condition $\mathcal{A}$, the resulting distribution $\mathbf{R} \,|\, \mathcal{A}$ is equivalent to the uniform random permutation $\mathbf{P}$.

Definition 15 (Distinguisher)**.** An adaptive distinguisher for $(\mathcal{X}, \mathcal{Y})$-random systems is defined as a $(\mathcal{Y}, \mathcal{X})$-random system $\mathbf{D}$ that interactively and adaptively queries $(\mathcal{X}, \mathcal{Y})$-random systems and ultimately outputs a bit D_q after some number of queries q. In $\mathbf{D}$ is a *non-adaptive* distinguisher, it must first fix its queries $X_1, \ldots, X_q$ in advance before receiving the outputs $Y_1, \ldots, Y_q$ and outputting its decision bit D_q.

The random experiment when we pair the distinguisher $\mathbf{D}$ with an $(\mathcal{X}, \mathcal{Y})$-random system $\mathbf{F}$ (where $\mathbf{D}$ submits a query to $\mathbf{F}$, $\mathbf{F}$ responds to $\mathbf{D}$, $\mathbf{D}$ submits another query, and so forth) is denoted by $\mathbf{D} \diamond \mathbf{F}$.

Definition 16 (Advantage)**.** We denote the advantage of a distinguisher $\mathbf{D}$ given q queries for distinguishing two $(\mathcal{X}, \mathcal{Y})$-random systems $\mathbf{F}$ and $\mathbf{G}$ as $\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G})$. There are several equivalent formal definitions. We can say that

$$\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) = \left|\mathsf{Pr}^{\mathbf{D}\diamond\mathbf{F}}[D_q = 1] - \mathsf{Pr}^{\mathbf{D}\diamond\mathbf{G}}[D_q = 1]\right|,$$

which requires that the decision bit D_q was computed optimally for this definition to be precise, or we can also say that the advantage is the statistical difference between the distributions $\mathbf{D} \diamond \mathbf{F}$ and $\mathbf{D} \diamond \mathbf{G}$ (which are distributions over the space $\mathcal{X}^q \times \mathcal{Y}^q$)

$$\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) = \frac{1}{2} \sum_{\mathcal{X}^q \times \mathcal{Y}^q} \left|\mathsf{Pr}^{\mathbf{D}\diamond\mathbf{F}}_{X^q Y^q} - \mathsf{Pr}^{\mathbf{D}\diamond\mathbf{G}}_{X^q Y^q}\right|.$$

The advantage of the best distinguisher on random systems $\mathbf{F}$ and $\mathbf{G}$ can be defined as

$$\Delta_q(\mathbf{F}, \mathbf{G}) = \max_{\mathbf{D}} \ \Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}).$$

On occasion we may want to restrict ourselves to only distinguishers $\mathbf{D}$ given certain resource constraints. Thus by $\Delta_{q,\mu}(\mathbf{F}, \mathbf{G})$ we mean the maximum advantage over all adversaries given q queries of total bit-length μ. If we wish to furthermore constrain their running time by t, we specify this as $\Delta_{t,q,\mu}(\mathbf{F}, \mathbf{G})$.

Going back to our earlier no-collision example, we have a random function $\mathbf{R}$ and the no-collision monotone condition $\mathcal{A}$ on $\mathbf{R}$. Recall that if $\mathbf{D}$ succeeds in causing A_q

to be false, then it will successfully distinguish $\mathbf{R}$ from $\mathbf{P}$ (because $\mathbf{P}$ does not have collisions). If this occurs, we say that $\mathbf{D}$ has *provoked* $\overline{A_q}$.

Definition 17 (Provoking Failure in a Monotone Condition)**.** Let $\mathbf{F}$ be a random system and let $\mathcal{A}$ be some monotone condition on $\mathbf{F}$. Denote the probability that $\mathbf{D}$ provokes the condition $\mathcal{A}$ to fail after q queries (that is, provokes $\overline{A_q}$) as

$$\nu_q^{\mathbf{D}}\left(\mathbf{D}, \overline{A_k}\right) = \mathsf{Pr}_{\overline{A_q}}^{\mathbf{D}\diamond\mathbf{F}} = 1 - \mathsf{Pr}_{A_q}^{\mathbf{D}\diamond\mathbf{F}}.$$

Denote the probability for the best such $\mathbf{D}$ to be

$$\nu_q\left(\mathbf{F}, \overline{A_k}\right) = \max_{\mathbf{D}} \ \nu_q^{\mathbf{D}}\left(\mathbf{D}, \overline{A_k}\right).$$

When we restrict ourselves to considering only non-adaptive distinguishers $\mathbf{D}$, we use the notation

$$\mu_q\left(\mathbf{F}, \overline{A_k}\right) = \max_{\text{non-adaptive } \mathbf{D}} \ \nu_q^{\mathbf{D}}\left(\mathbf{D}, \overline{A_k}\right).$$

In the prior no-collision example, if $\mathbf{D}$ is some algorithm for finding a collision in the random function $\mathbf{R}$, then clearly its success probability is bounded from above by $\nu_q\left(\mathbf{F}, \overline{A_k}\right)$.

4.1.3 Bounding Distinguishability

Throughout this chapter we will make use of some important lemmas due to Maurer [21], the proofs of which we will omit for the sake of brevity, as we focus on our original results. However, we will give proof sketches of Maurer's lemmas when it is helpful to do so. For a much more rigorous treatment of the theory of random systems, we refer the reader to Maurer and Pietrzak [21, 30].

Lemma 4.1 (Lemma 5(i) [21])**.** For random systems, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$,

$$\Delta_q(\mathbf{F}, \mathbf{H}) \leq \Delta_q(\mathbf{F}, \mathbf{G}) + \Delta_q(\mathbf{G}, \mathbf{H}).$$

PROOF. This follows directly by the triangle inequality. □

In the following lemma we show an upper bound on the advantage of a distinguisher $\mathbf{D}$ for random systems $\mathbf{F}$ and $\mathbf{G}$ based on the ability of $\mathbf{D}$ to provoke the failure of a monotone condition on $\mathbf{F}$.

Lemma 4.2 (Theorem 1(i) [21]). For random systems $\mathbf{F}$ and $\mathbf{G}$, if $\mathcal{A}$ is some monotone condition such that $\mathbf{F} \,|\, \mathcal{A} \equiv \mathbf{G}$, then the advantage of the best distinguisher for $\mathbf{F}$ and $\mathbf{G}$ is bounded by the probability of provoking $\overline{\mathcal{A}}$ given the best adaptive strategy.

$$\Delta_q(\mathbf{F}, \mathbf{G}) \leq \nu_q\big(\mathbf{F}, \overline{A_k}\big)$$

PROOF SKETCH.

$$\begin{aligned}
\Delta_q(\mathbf{F}, \mathbf{G}) &\leq \nu_q\big(\mathbf{F}, \overline{A_k}\big) \cdot \Delta_q\big(\mathbf{F} \,|\, \overline{\mathcal{A}}, \mathbf{G}\big) + \big(1 - \nu_q\big(\mathbf{F}, \overline{A_k}\big)\big) \cdot \Delta_q\big(\mathbf{F} \,|\, \mathcal{A}, \mathbf{G}\big) \\
&\leq \nu_q\big(\mathbf{F}, \overline{A_k}\big) \cdot 1 + \big(1 - \nu_q\big(\mathbf{F}, \overline{A_k}\big)\big) \cdot 0 \\
&= \nu_q\big(\mathbf{F}, \overline{A_k}\big)
\end{aligned}$$

The first inequality holds by the law of total probability. The second is due to the fact that $\mathbf{F} \,|\, \mathcal{A} \equiv \mathbf{G}$, hence $\Delta_q\big(\mathbf{F} \,|\, \mathcal{A}, \mathbf{G}\big) = 0$. □

In certain situations, adaptivity does not increase a distinguisher's advantage with respect to provoking the failure of some monotone condition. The following lemma provides a sufficient condition for this to be true.

Lemma 4.3 (Theorem 2 [21]). For a random system $\mathbf{F}$ with a monotone condition $\mathcal{A}$, if there exists a random system $\mathbf{G}$ such that $\mathbf{F} \,|\, \mathcal{A} \equiv \mathbf{G}$, i.e.

$$\text{for all } i \geq 1, \mathsf{Pr}^{\mathbf{F}}_{Y^i|X^i A_i} \equiv \mathsf{Pr}^{\mathbf{G}}_{Y^i|X^i}$$

then adaptivity does not help in provoking $\overline{A_q}$:

$$\nu_q\big(\mathbf{F}, \overline{A_k}\big) = \mu_q\big(\mathbf{F}, \overline{A_k}\big)\,.$$

Proof. See [30, Lemma 6]. □

To illustrate the usefulness of this framework, we return to our demonstration of the indistinguishability of a random function $\mathbf{R} : \{0,1\}^n \to \{0,1\}^n$ from a random permutation $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$ by any distinguisher $\mathbf{D}$ asking $o\big(2^{\frac{n}{2}}\big)$ queries (adapted from [30, Examples 1–3]).

Example 2. We first begin by defining the monotone condition $\mathcal{A} = \{A_i\}$, where A_i is the event that after the i^{th} query all distinct inputs have produced distinct outputs. It is fairly straightforward to demonstrate that $\mathbf{R}\,|\,\mathcal{A} \equiv \mathbf{P}$: unless one has observed a collision, a random function has output distribution identical to a random permutation.

Therefore by Lemma 4.2, $\Delta_q(\mathbf{R},\mathbf{P}) \le \nu_q\big(\mathbf{R},\overline{A_k}\big)$. By definition, $\nu_q\big(\mathbf{R},\overline{A_k}\big)$ is the probability of success of the best distinguisher to provoke a collision (using distinct inputs) on a uniformly random function $\mathbf{R}$, which is clearly bounded by the Birthday Paradox (see Appendix A).

$$\Delta_q(\mathbf{R},\mathbf{P}) \le \nu_q\big(\mathbf{R},\overline{A_k}\big) \le \frac{q(q-1)}{2^{n+1}}$$

For $q = o\big(2^{\frac{n}{2}}\big)$, this is a negligible probability of success.

4.2 MD6 as a Domain Extender for FIL-PRFs

With the random system framework and the above lemmas, we can now prove that MD6 behaves as a domain extender on FIL-PRFs that preserves pseudorandomness.

4.2.1 Preliminaries

Here we define some random systems used throughout this section.

- Let $\mathbf{R}$ denote the random function with uniform distribution over functions mapping $\{0,1\}^n \to \{0,1\}^c$

- Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$ be the compression function used. Then let $\mathbf{F} : \{0,1\}^n \to \{0,1\}^c$ denote the random function with a uniform distribution over the set
$$\left\{ f(K, \cdot) \mid K \in \{0,1\}^k \right\}.$$

- Let $\mathbf{O}$ denote a random function with output space $\mathcal{Y} = \{0,1\}^d$ and input space $\mathcal{X} = \{0,1\}^*$ such that for all $i \geq 1, x \in \{0,1\}^*, y \in \{0,1\}^d$,
$$\mathsf{Pr}^{\mathbf{O}}_{Y_i|X_i}(y, x) = \frac{1}{2^d}.$$
$\mathbf{O}$ is usually referred to as a *random oracle*.

- For a random function $\mathbf{G} : \{0,1\}^n \to \{0,1\}^c$, let $\mathcal{H}^{\mathbf{G}}$ denote the random function mapping $\{0,1\}^* \to \{0,1\}^n$ by applying the MD6 mode of operation (without the final compression function and chop) with $\mathbf{G}$ as the compression function.

- For a random function $\mathbf{G} : \{0,1\}^n \to \{0,1\}^c$, let $\mathcal{M}^{\mathbf{G}}$ denote the random function mapping $\{0,1\}^* \to \{0,1\}^d$ by applying the MD6 mode of operation with $\mathbf{G}$ as the compression function. Thus
$$\mathcal{M}^{\mathbf{G}} = chop_d \circ \mathbf{G} \circ \mathcal{H}^{\mathbf{G}}.$$

Our goal in this section will be to demonstrate that if $\mathbf{F}$ is a FIL-PRF (indistinguishable from $\mathbf{R}$), then $\mathcal{M}^{\mathbf{F}}$ will be a VIL-PRF (indistinguishable from $\mathbf{O}$).

Definition 18 (FIL-Pseudorandom Function)**.** A random function $\mathbf{G} : \{0,1\}^n \to \{0,1\}^c$ is a (t, q, ε)-secure FIL-pseudorandom function (FIL-PRF) if it is (t, q, ε)-indistinguishable from the uniform random function on the same domain and range,
$$\Delta_{t,q}(\mathbf{G}, \mathbf{R}) \leq \varepsilon.$$

We say that $\mathbf{G}$ is a (q, ε)-secure FIL-quasirandom function (FIL-QRF) if it is a (∞, q, ε)-secure FIL-PRF. That is, we do not restrict the computational abilities of the distinguisher $\mathbf{D}$ but we restrict the number of queries that it can make. As we are considering distinguishers unconstrained by time, we omit this variable from the subscript, and write

$$\Delta_q(\mathbf{G}, \mathbf{R}) \leq \varepsilon.$$

Therefore if $\mathbf{G}$ is a (q, ε)-secure FIL-QRF, then it is a (t, q, ε)-secure FIL-PRF for any choice of t.

Definition 19 (VIL-Pseudorandom Function)**.** A random function $\mathbf{G} : \{0,1\}^* \to \{0,1\}^d$ is a (t, q, μ, ε)-secure VIL-pseudorandom function (VIL-PRF) if it is (t, q, μ, ε)-indistinguishable from a random oracle $\mathbf{O}$,

$$\Delta_{t,q,\mu}(\mathbf{G}, \mathbf{O}) \leq \varepsilon.$$

As above, we say that $\mathbf{G}$ is a (q, μ, ε)-secure VIL-quasirandom function (VIL-QRF) if

$$\Delta_{q,\mu}(\mathbf{G}, \mathbf{O}) \leq \varepsilon.$$

4.2.2 Indistinguishability

Our proof that $\mathcal{M}^{\mathbf{F}}$ is indistinguishable from $\mathbf{O}$ proceeds as follows. First we notice by the triangle inequality (Lemma 4.1) that

$$\Delta_{q,\mu}\big(\mathcal{M}^{\mathbf{F}}, \mathbf{O}\big) \leq \Delta_{q,\mu}\big(\mathcal{M}^{\mathbf{F}}, \mathcal{M}^{\mathbf{R}}\big) + \Delta_{q,\mu}\big(\mathcal{M}^{\mathbf{R}}, \mathbf{O}\big)\,. \tag{4.1}$$

Then it only remains to bound the quantities on the right-hand side. In Lemma 4.4, we show how to bound the first term by the advantage for distinguishing between $\mathbf{F}$ and $\mathbf{R}$ (which by assumption is small).

Lemma 4.4 (Adapted from Lemma 5(ii) [21])**.** For random systems $\mathbf{G}$ and $\mathbf{H}$ that map $\{0,1\}^n \to \{0,1\}^c$,

$$\Delta_{t',q,\mu}\big(\mathcal{M}^{\mathbf{G}}, \mathcal{M}^{\mathbf{H}}\big) \leq \Delta_{t,\delta(q,\mu)}(\mathbf{G}, \mathbf{H})\,,$$

where $t' = t - O(\delta(q, \mu))$.

Proof. Given a distinguisher $\mathbf{D}$ for $\mathcal{M}^{\mathbf{G}}$ and $\mathcal{M}^{\mathbf{H}}$ with resource constraints t', q, and μ, we can construct an algorithm for distinguishing $\mathbf{G}$ and $\mathbf{H}$ that uses $\mathbf{D}$ as a subroutine. This algorithm responds to the queries of $\mathbf{D}$ by simulating the MD6 mode of operation $\mathcal{M}$ on $\mathbf{G}$ or $\mathbf{H}$. By Lemma 2.1, this requires at most $\delta(q, \mu)$ queries. Since the only work to be done is in simulating $\mathcal{M}$, this takes total time $t' + O(\delta(q, \mu))$. □

For the second term of Equation (4.1), we proceed in a manner that is similar to the proof that a random function is indistinguishable from a random permutation (Example 2). That is, we will construct a monotone condition $\mathcal{A}$ such that $\mathcal{M}^{\mathbf{R}} \,\big|\, \mathcal{A} \equiv \mathbf{O}$, and we then bound the probability of provoking $\overline{\mathcal{A}}$. However, unlike a random permutation, a random oracle $\mathbf{O}$ naturally has collisions, so our previous no-collision monotone condition is not applicable. Thus we design a new monotone condition, one that involves so-called *bad collisions*. A bad collision in $\mathcal{M}^{\mathbf{R}}$ is one that occurs prior to the final compression function in the MD6 mode of operation. As we will show in Lemma 4.5, if we condition on the absence of bad collisions in $\mathcal{M}^{\mathbf{R}}$, then the distribution of its outputs are identical to that of $\mathbf{O}$.

Definition 20 (Bad Collision)**.** For a random function $\mathbf{G} : \{0,1\}^n \to \{0,1\}^c$ and messages $M, M' \in \{0,1\}^*$, let $\mathrm{BC}_{\mathcal{M}_L^{\mathbf{G}}}(M, M')$ denote the event that

$$M \neq M', \quad \text{and} \quad \mathcal{H}_L^{\mathbf{G}}(M) = \mathcal{H}_L^{\mathbf{G}}(M').$$

Note that a bad collision necessarily implies a collision in $\mathcal{M}^{\mathbf{F}}$.

A bad collision on M and M' means that not only does $\mathcal{H}_L^{\mathbf{G}}(M) = \mathcal{H}_L^{\mathbf{G}}(M')$, but $\mathit{height}_L(|M|) = \mathit{height}_L(|M'|)$ as well. This is because one word of the input to each compression function is devoted to U, the representation of the level ℓ and index i of the compression function in the hash tree. In particular, for the final compression function, the height of the hash tree is encoded into U, since $\ell + i - 2$ is the height of the tree. Therefore if $height_L(|M|) \neq height_L(|M'|)$ then it is impossible for $\mathcal{H}_L^{\mathbf{G}}(M) = \mathcal{H}_L^{\mathbf{G}}(M')$, because this height information is encoded into the output of $\mathcal{H}_L^{\mathbf{G}}$.

Lemma 4.5. Let $\mathcal{A} = \{A_i\}$ be the monotone condition on $\mathcal{M}_L^{\mathbf{R}}$ such that A_i is the event that there are no bad collisions in the first i queries $(M_1, M_2, \ldots, M_i)$:

$$A_i = \bigwedge_{1 \leq j \leq j' \leq i} \overline{\mathrm{BC}_{\mathcal{M}_L^{\mathbf{R}}}(M_j, M_{j'})}.$$

Then $\mathcal{M}_L^{\mathbf{R}} \,\big|\, \mathcal{A} \equiv \mathbf{O}$.

Proof. Recall that $\mathcal{M}_L^{\mathbf{R}} = chop_d \circ \mathbf{R} \circ \mathcal{H}_L^{\mathbf{R}}$, and further, the event $\overline{\mathrm{BC}_{\mathcal{M}_L^{\mathbf{R}}}(M, M')}$ implies that $\mathcal{H}_L^{\mathbf{R}}(M) \neq \mathcal{H}_L^{\mathbf{R}}(M')$. So, effectively, we are asking about the output distribution of $(chop_d \circ \mathbf{R}) : \{0,1\}^n \to \{0,1\}^d$, conditioned on having distinct inputs, as compared to the output distribution of $\mathbf{O} : \{0,1\}^* \to \{0,1\}^d$, also conditioned on having distinct inputs. Since $chop_d \circ \mathbf{R}$ is just the random function with uniform distribution over all functions mapping $\{0,1\}^n \to \{0,1\}^d$, these distributions are identical (namely, the uniform distribution over $\{0,1\}^d$). $\square$

We can now apply the above lemma to bound the probability of distinguishing between $\mathcal{M}_L^{\mathbf{R}}$ and $\mathbf{O}$.

Lemma 4.6. If $\mathcal{M}_L^{\mathbf{R}} \,\big|\, \mathcal{A} \equiv \mathbf{O}$, then

$$\Delta_q\big(\mathcal{M}_L^{\mathbf{R}}, \mathbf{O}\big) \leq \frac{q(q-1)}{2} \cdot \frac{height_L\left(\frac{\mu}{q}\right)}{2^c}.$$

Proof. By Lemma 4.2, since $\mathcal{M}^{\mathbf{R}} \,\big|\, \mathcal{A} \equiv \mathbf{O}$, any adversary's advantage for distinguishing between $\mathcal{M}^{\mathbf{R}}$ and $\mathbf{O}$ with q queries is bounded by the probability of success by the best adaptive algorithm at provoking $\overline{A_q}$ (that is, finding a bad collision).

$$\Delta_q\big(\mathcal{M}^{\mathbf{R}}, \mathbf{O}\big) \leq \nu_q\big(\mathcal{M}^{\mathbf{R}}, \overline{A_k}\big)\,.$$

In addition, by Lemma 4.3, the probability of success for the best adaptive strategy is no better than that for the best non-adaptive strategy (and in fact, they are equal). Therefore,

$$\nu_q(\mathcal{M}^{\mathbf{R}}, \overline{A_k}) = \mu_q(\mathcal{M}^{\mathbf{R}}, \overline{A_k}) \,.$$

Thus, we may simply consider the probability of provoking a bad collision non-adaptively, which is the great benefit of using Maurer's framework. To do this, we will first compute the probability that two arbitrary distinct messages M and M' will have a bad collision, and then apply the birthday bound (see Appendix A).

Recall that a bad collision occurs when we have two messages $M \neq M'$ where $height_L(M) = height_L(M')$ and $\mathcal{H}^{\mathbf{R}}(M) = \mathcal{H}^{\mathbf{R}}(M')$. Therefore we can first restrict ourselves to considering only messages M, M' which have equal height h.

Now, with this assumption, we wish to upper bound $\Pr[\mathrm{BC}_{\mathcal{M}^{\mathbf{R}}}(M, M')]$. Our goal will be to prove by induction on the height h that

$$\Pr[\mathrm{BC}_{\mathcal{M}^{\mathbf{R}}}(M, M')] \leq \frac{h}{2^c}.$$

The base case here is straightforward. It is impossible for two trees of height 1 to have a bad collision. This is because height 1 trees have only one compression function, namely the final one. By definition, a bad collision is a collision that occurs before the final compression function, which simultaneously requires that $M \neq M'$ and $M = M'$. Thus we have a contradiction.

For the inductive step, note that the output of $\mathcal{H}^{\mathbf{R}}(M)$ can be described as follows. First, there exists a partition of M into $M_1 \| M_2 \| M_3 \| M_4$ of four (some possibly empty) bit strings such that

$$\mathcal{H}^{\mathbf{R}}(M) = U \| V \| H_1^{\mathbf{R}}(M_1) \| H_2^{\mathbf{R}}(M_2) \| H_3^{\mathbf{R}}(M_3) \| H_4^{\mathbf{R}}(M_4).$$

Here the $H_i^{\mathbf{R}}$ are random functions mapping $\{0,1\}^* \to \{0,1\}^c$ and U and V are the auxiliary control information inserted by the MD6 mode of operation. In particular, when viewed as a hash tree, the height of each $H_i^{\mathbf{R}}(M_i)$ is at most $h-1$. In addition, if we also partition M' in this fashion, we notice that since $M \neq M'$ there exists a j such that $M_j \neq M'_j$. Therefore

$$\begin{aligned}
\Pr[\mathrm{BC}_{\mathcal{M}^{\mathbf{R}}}(M, M')] &= \Pr\left[\mathcal{H}^{\mathbf{R}}(M) = \mathcal{H}^{\mathbf{R}}(M')\right] \\
&\leq \Pr\left[\bigwedge_{i=1}^{4} \left(H_i^{\mathbf{R}}(M_i) = H_i^{\mathbf{R}}(M'_i)\right)\right] \\
&= \prod_{i=1}^{4} \Pr\left[H_i^{\mathbf{R}}(M_i) = H_i^{\mathbf{R}}(M'_i)\right] \\
&\leq \Pr\left[H_j^{\mathbf{R}}(M_j) = H_j^{\mathbf{R}}(M'_j)\right].
\end{aligned}$$

Thus, we've upper bounded the probability of a bad collision occurring by the probability of there being a collision for one of the $H_i^{\mathbf{R}}$. Let E denote the event that this collision, $H_j^{\mathbf{R}}(M_j) = H_j^{\mathbf{R}}(M'_j)$, occurs. Then

$$\begin{aligned}
\Pr[E] = \Pr&\left[E \,\middle|\, \mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)\right] \cdot \Pr\left[\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)\right] \\
&+ \Pr\left[E \,\middle|\, \overline{\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)}\right] \cdot \Pr\left[\overline{\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)}\right] && (4.2) \\
&\leq 1 \cdot \Pr\left[\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)\right] + \frac{1}{2^c} \cdot \Pr\left[\overline{\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)}\right] && (4.3) \\
&\leq \Pr\left[\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)\right] + \frac{1}{2^c} \\
&\leq \frac{h-1}{2^c} + \frac{1}{2^c} && (4.4) \\
&= \frac{h}{2^c} && (4.5)
\end{aligned}$$

Equation (4.2) follows by the law of total probability. In the derivation of Inequality (4.3), we use the fact that the conditional probability $\Pr\left[E \,\middle|\, \overline{\mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)}\right]$ is just the probability of collision for the random function $\mathbf{R}$ on distinct inputs, which is equal to $\frac{1}{2^c}$. In addition, $\Pr\left[E \,\middle|\, \mathrm{BC}_{H_j^{\mathbf{R}}}(M_j, M'_j)\right]$ is the probability of collision for the random function $\mathbf{R}$ on identical inputs, which is just 1. For Inequality (4.4), we can use our inductive hypothesis, since as we noted earlier $H_j^{\mathbf{R}}(M_j)$ has height at most

$h - 1$, thereby arriving at Inequality (4.5).

Since we've shown that for any two messages M and M' the probability of a bad collision is bounded by $\frac{h}{2^c}$, it remains to bound the total probability of causing a bad collision given q non-adaptive queries of total length at most μ. Since an adversary cannot cause a bad collision by querying two messages with different heights, the best strategy for the adversary is to produce q queries of all equal height $height_L\left(\frac{\mu}{q}\right)$, and by the birthday bound

$$\Delta_q\big(\mathcal{M}^{\mathbf{R}}, \mathbf{O}\big) \leq \frac{q(q-1)}{2} \cdot \frac{height_L\left(\frac{\mu}{q}\right)}{2^c}.$$

□

Now that we have bounded the probability of success for any distinguisher given resources q and μ, we show that MD6 acts as a domain extender for FIL-QRFs, and later we will show that it does the same for FIL-PRFs.

Theorem 4.7. Fix the resource constraints q, and μ. If $\mathbf{F}$ is a $(\delta(q,\mu), \varepsilon)$-secure FIL-QRF, then $\mathcal{M}_L^{\mathbf{F}}$ is a $(q, \mu, \varepsilon + \beta(q,\mu))$-secure VIL-QRF, where

$$\beta(q,\mu) = \frac{q(q-1)}{2} \cdot \frac{height_L\left(\frac{\mu}{q}\right)}{2^c}.$$

Proof. By Lemma 4.1 (the triangle inequality), the advantage of any adversary given q queries and total message bit-length μ in distinguishing $\mathcal{M}^{\mathbf{F}}$ from $\mathbf{O}$ is

$$\Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathbf{O}\big) \leq \Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathcal{M}_L^{\mathbf{R}}\big) + \Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{R}}, \mathbf{O}\big).$$

By Lemma 4.4, the advantage of any adversary given q queries and μ bits in distinguishing $\mathcal{M}_L^{\mathbf{F}}$ and $\mathcal{M}_L^{\mathbf{R}}$ is

$$\Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathcal{M}_L^{\mathbf{R}}\big) \leq \Delta_{\delta(q,\mu)}(\mathbf{F}, \mathbf{R}) \leq \varepsilon.$$

Therefore it remains only to bound $\Delta_q\big(\mathcal{M}^{\mathbf{R}}, \mathbf{O}\big)$. By applying Lemmas 4.5 and 4.6, we can bound this by

$$\Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{R}}, \mathbf{O}\big) \leq \frac{q(q-1)}{2} \cdot \frac{height_L\left(\frac{\mu}{q}\right)}{2^c}.$$

Finally, combining this with our previous results yields

$$\Delta_{q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathbf{O}\big) \leq \varepsilon + \frac{q(q-1)}{2} \cdot \frac{height_L\left(\frac{\mu}{q}\right)}{2^c}.$$

□

Corollary 4.8. If $\mathbf{F}$ is a $(t, \delta(q,\mu), \varepsilon)$-secure FIL-PRF, then $\mathcal{M}_L^{\mathbf{F}}$ is a $(t', q, \mu, \varepsilon + \beta(q,\mu))$-secure VIL-PRF, where $t' = t - O(\delta(q,\mu))$.

Proof. The proof proceeds almost identically to Theorem 4.7.

$$\begin{aligned}
\Delta_{t',q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathbf{O}\big) &\leq \Delta_{t',q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathcal{M}_L^{\mathbf{R}}\big) + \Delta_{t',q,\mu}\big(\mathcal{M}_L^{\mathbf{R}}, \mathbf{O}\big) \\
&\leq \Delta_{t',q,\mu}\big(\mathcal{M}_L^{\mathbf{F}}, \mathcal{M}_L^{\mathbf{R}}\big) + \beta(q,\mu) \\
&\leq \Delta_{t,\delta(q,\mu)}(\mathbf{F}, \mathbf{R}) + \beta(q,\mu) \qquad (4.6) \\
&\leq \varepsilon + \beta(q,\mu)
\end{aligned}$$

Inequality (4.6) follows as a corollary to Lemma 4.4, since simulating the query responses to the adversary takes time $O(\delta(q,\mu))$. □

One important note is that for large values of L, $height_L(x)$ grows logarithmically in x. However, for the iterative mode of operation with $L = 0$, $height_L(x)$ grows linearly in x. Holding q fixed, this is asymptotically the same as the bound shown by Bellare et al. for the Cipher Block Chaining mode of operation [7], which was approximately $O\left(\frac{\ell q^2}{2^c}\right)$, where ℓ is the block length of the longest query. Therefore the MD6 mode of operation for large L represents an asymptotic improvement (logarithmic in $height_L(\mu/q)$ as opposed to linear) over the Cipher Block Chaining mode of operation.

Chapter 5

Unpredictability (MACs)

That a family of functions has the property of pseudorandomness is a very strong assumption to make. Much simpler is the assumption that a family of functions is merely *unpredictable*, which allows it to function as a MAC (recall that pseudorandomness implies unpredictability, and thus PRFs are also MACs [18, 19].) However, if one simply wants to prove that some hash function H^f is a secure MAC, it seems unnecessary to derive this property by assuming that the underlying compression f is pseudorandom. One might wonder whether it is instead possible to prove that H^f is a MAC if f is a MAC, analogous to the proof in Section 4.2. If so, it might alleviate some concerns about the use of a hash function H^f as a MAC, since a successful pseudorandomness distinguisher for f would not necessarily invalidate the MAC capabilities of H^f (unless f is also shown to be predictable as well).

This may not be true if H is the Cipher Block Chaining (CBC) mode of operation, H_{CBC}. Whereas one can prove concretely that CBC-MAC is pseudorandom if its underlying block cipher is pseudorandom [6, 7], An and Bellare constructed a simple FIL-MAC f such that H^f_{CBC} does not share the property of unpredictability [1]. However, they also demonstrated that a two-keyed variant of the Merkle-Damgård construction is a VIL-MAC if its compression function is a FIL-MAC. Thus it is interesting for us to consider if MD6 also is a domain extender that preserves the property of unpredictability.

5.1 Preliminaries

We begin with some definitions that will be essential to our later proofs.

Definition 21 (FIL-MAC)**.** For a keyed FIL function $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$, define the advantage of an adversary A for forging a MAC as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{fil-mac}} = \Pr\left[\begin{array}{c} K \xleftarrow{\$} \{0,1\}^k; \\ (m, D) \leftarrow \mathsf{A}^{f(K,\cdot)} \end{array} : \begin{array}{c} f(K,m) = D, \\ m \text{ was not a query to } f(K,\cdot) \end{array}\right]$$

We define the insecurity of the FIL-MAC f to be

$$\mathbf{InSec}_{f}^{\text{fil-mac}}(t,q) = \max_{\mathsf{A}} \left\{\mathbf{Adv}_{\mathsf{A}}^{\text{fil-mac}}\right\},$$

where the maximum is taken over all adversaries A with running time t and number of $f(K,\cdot)$ oracle queries q.

Definition 22 (VIL-MAC)**.** For a keyed VIL function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^d$, define the advantage of an adversary A for forging a MAC as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{vil-mac}} = \Pr\left[\begin{array}{c} K \xleftarrow{\$} \{0,1\}^k; \\ (M, D) \leftarrow \mathsf{A}^{H(K,\cdot)} \end{array} : \begin{array}{c} H(K,M) = D, \\ M \text{ was not a query to } H(K,\cdot) \end{array}\right]$$

We define the insecurity of the VIL-MAC H to be

$$\mathbf{InSec}_{H}^{\text{vil-mac}}(t,q,\mu) = \max_{\mathsf{A}} \left\{\mathbf{Adv}_{\mathsf{A}}^{\text{vil-mac}}\right\},$$

where the maximum is taken over all adversaries A with running time t and number of $H(K,\cdot)$ oracle queries q and total query bit-length μ.

To proceed, we must define the notion of *weak collision resistance*. As the name implies, this is a weaker notion of the collision resistance defined in Chapter 3; rather than giving the collision-finding algorithm the key K, we instead give it only oracle access to the keyed function.

Definition 23 (FIL-WCR)**.** For a keyed FIL function $f : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^c$, define the advantage of an adversary A for FIL-weak collision resistance as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{fil-wcr}} = \Pr\left[\begin{array}{ccc} K \xleftarrow{\$} \{0,1\}^k; & & m \neq m' \\ (m, m') \leftarrow \mathsf{A}^{f(K,\cdot)} & \mathbf{:} & f(K,m) = f(K,m'), \end{array}\right]$$

We define the insecurity of the FIL-WCR f to be

$$\mathbf{InSec}_f^{\text{fil-wcr}}(t,q) = \max_{\mathsf{A}}\left\{\mathbf{Adv}_{\mathsf{A}}^{\text{fil-wcr}}\right\},$$

where the maximum is taken over all adversaries A with running time t and number of $f(K,\cdot)$ oracle queries q.

Definition 24 (VIL-WCR)**.** For a keyed VIL function $H : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^c$, define the advantage of an adversary A for VIL-weak collision resistance as

$$\mathbf{Adv}_{\mathsf{A}}^{\text{vil-wcr}} = \Pr\left[\begin{array}{ccc} K \xleftarrow{\$} \{0,1\}^k; & & M \neq M', \\ (M, M') \leftarrow \mathsf{A}^{H(K,\cdot)} & \mathbf{:} & H(K,M) = H(K,M') \end{array}\right]$$

We define the insecurity of the VIL-WCR H to be

$$\mathbf{InSec}_H^{\text{vil-wcr}}(t,q,\mu) = \max_{\mathsf{A}}\left\{\mathbf{Adv}_{\mathsf{A}}^{\text{vil-wcr}}\right\},$$

where the maximum is taken over all adversaries A with running time t and number of $H(K,\cdot)$ oracle queries q and total query bit-length μ.

We also make use of some lemmas from An and Bellare. We restate them here, without rigorous proofs, which can be found in [1].

5.1.1 Important Lemmas

The two-keyed variant of Merkle-Damgård shown An and Bellare is defined as follows. Given a keyed compression function $f : \{0,1\}^k \times \{0,1\}^{\ell+b} \rightarrow \{0,1\}^{\ell}$ and two keys K_1 and K_2, use the strengthened Merkle-Damgård construction where all compression functions except for the *final* one use key K_1. The final compression function then uses K_2. In their proof that this is a domain extender for FIL-MACs, An and Bellare used several steps.

1. Show that a FIL-MAC g is also FIL-weak collision resistant.

2. Prove that the Merkle-Damgård construction (without the last compression function) is a domain extender for FIL-weak collision resistance. That is, if the compression function g used is FIL-WCR, then the overall hash function h (without the last compression function) is VIL-WCR.

3. Demonstrate that composing a FIL-MAC g with a VIL-WCR function h (with independent keys) is a VIL-MAC. Therefore by points 1 and 2, we have a VIL-MAC from a FIL-MAC.

We begin with a formal statement of the last point.

Lemma 5.1 (Lemma 4.2 [1]). Let $g : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^d$ be a FIL-MAC and let $h : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a VIL-WCR function. Define $H : \{0,1\}^{2k} \times \{0,1\}^* \to \{0,1\}^d$ as

$$H(K_1, K_2, M) = f(K_2, h(K_1, M))$$

for keys $K_1, K_2 \in \{0,1\}^k$ and $M \in \{0,1\}^*$. Then H is a VIL-MAC with

$$\mathbf{InSec}_H^{\text{vil-mac}}(t, q, \mu) \leq \mathbf{InSec}_g^{\text{fil-mac}}(t, q) + \mathbf{InSec}_h^{\text{vil-wcr}}(t, q, \mu).$$

Proof. See [1, Appendix A.1]. □

In addition, we will also make use of the following lemma, which states formally point 1 from above.

Lemma 5.2 (Lemma 4.4 [1]). Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$ be a FIL-MAC family of functions. Then it is also a FIL-WCR family with

$$\mathbf{InSec}_f^{\text{fil-wcr}}(t, q) \leq \frac{q(q-1)}{2} \cdot \mathbf{InSec}_f^{\text{fil-mac}}(t + O(q), q).$$

PROOF. See [1, Appendix A.3]. □

Since the MD6 mode of operation is different from the two-keyed Merkle-Damgård construction of An and Bellare, we omit the formal statement for point 2. Instead, we will prove our own version in Lemma 5.3 for the MD6 mode of operation.

5.1.2 A Two-Keyed Variant of MD6

Note that in Lemma 5.1 the hash function H has a keyspace with twice as many bits as the underlying functions g and h. Therefore, a straight adaptation of the techniques of An and Bellare [1] for MD6 does not immediately follow, as MD6 has only a single key. Thus, we demonstrate here that a two-keyed variant of MD6 is a domain extender for FIL-MACs, in much the same fashion as An and Bellare's approach. That is, we show that the function

$$\mathcal{M}^f[2](K_1, K_2, M) = \mathit{chop}_d(f(K_2, \mathcal{H}^f(K_1, M)))$$

where $\mathcal{M}^f[2] : \{0,1\}^{2k} \times \{0,1\}^* \to \{0,1\}^d$ is a VIL-MAC if f and $\mathit{chop}_d \circ f$ are FIL-MACs.

We begin by proving that if the compression function f is a FIL-WCR, then $\mathcal{H}^f$ is a VIL-WCR function. This is analogous to [1, Lemma 4.3], where An and Bellare prove that the Merkle-Damgård construction also acts as a domain extender for FIL-WCRs. The proof of this lemma is very similar to the proof of Theorem 3.4, as weak collision resistance is a weaker notion of the standard collision resistance. However, due to the additional query resource constraints in the weak collision resistance definition, we must be precise in our adaptation.

Lemma 5.3. Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$ be a FIL-WCR family of functions. Then $\mathcal{H}^f : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ is a VIL-WCR family of functions with

$$\mathbf{InSec}^{\text{vil-wcr}}_{\mathcal{H}^f}(t, q, \mu) \leq \mathbf{InSec}^{\text{fil-wcr}}_{f}(2t, \delta(q, \mu)).$$

PROOF. We proceed in a manner that is very similar to the proof for Theorem 3.4. Let A be an algorithm with the best possible success for breaking the VIL-weak collision resistance of $\mathcal{H}^f$, given resources t, q and μ. As before, we construct an algorithm C that uses A as a subroutine to attack the FIL-weak collision resistance of f.

Recall that in the weak collision resistance setting, instead of being given the key K as input, we are instead only given oracle access to the function keyed by K. Therefore $\mathsf{C}^{f(K,\cdot)}$ is given time $2t$, q queries, and μ total bits queried, and must produce messages $m \neq m'$ such that $f(K, m) = f(K, m')$.

```
Algorithm $\mathsf{C}^{f(K,\cdot)}$
Input: $\mathsf{A}$, the $\mathcal{H}^f$ collision-finding algorithm
Output: $m \neq m'$, such that $f(K,m) = f(K,m')$
1.  for $i \leq 1$ to $q$
2.      do $\mathsf{A} \rightarrow M_i$,
3.          $\mathsf{A} \leftarrow \mathcal{H}^{f(K,\cdot)}(M_i)$
4.  $\mathsf{A} \rightarrow (M, M')$
5.  $P \leftarrow \mathcal{H}^{f(K,\cdot)}(M)$ and store each input to $f(K,\cdot)$
6.  $P' \leftarrow \mathcal{H}^{f(K,\cdot)}(M')$ and store each input to $f(K,\cdot)$
7.  if $M \neq M'$ and $P = P'$
8.     then Use Theorem 3.3 to find two messages $m \neq m'$ s.t. $f(K,m) = f(K,m')$
9.          return $(m, m')$
10.    else return $(0^n, 0^n)$
```

Note that although we are using $\mathcal{H}^f$ instead of $\mathcal{M}^f$, Theorem 3.3 still applies. This is because $M \neq M'$ and $\mathcal{M}^f(K, M) = \mathcal{M}^f(K, M')$, and thus a collision in f exists. However, it does not occur during the last compression function f, since $P = P'$. Therefore it must be "contained in" $\mathcal{H}^f$.

The advantage for C for breaking the weak collision resistance of f is at least the advantage of A for breaking the weak collision resistance of $\mathcal{H}^f$, because C succeeds precisely when A succeeds. As in Theorem 3.4, the amount of required is at most $2t$

(the time it takes to run A plus the time it takes to perform the hashes). In addition, by Lemma 2.1, the total number of oracle queries to $f(K,\cdot)$ that we need to make is bounded by $\delta(q,\mu)$. $\square$

We can now prove that the two-keyed variant of MD6 $\mathcal{M}^f[2]$ is a VIL-MAC, assuming that f is a FIL-MAC and also that $g = chop_d \circ f$ is a FIL-MAC. Note that the fact that g is a FIL-MAC does not follow automatically from the fact that f is a FIL-MAC, so we will treat them as functions with separate levels of security.

Theorem 5.4. Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$ be a FIL-MAC. Define the final compression function $g = (chop_d \circ f) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^d$ and suppose that it is also a FIL-MAC. Then the two-keyed variant of MD6 $\mathcal{M}^f[2] : \{0,1\}^{2k} \times \{0,1\}^* \to \{0,1\}^d$ is a VIL-MAC with

$$\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t,q,\mu) \le \mathbf{InSec}^{\text{fil-mac}}_{g}(t,q) + \frac{\delta(q,\mu)^2}{2} \cdot \mathbf{InSec}^{\text{fil-mac}}_{f}\left(2t + O(\delta(q,\mu)), \delta(q,\mu)\right).$$

Proof. As in [1, Theorem 4.1], the proof of this theorem follows from Lemmas 5.1, 5.2 and 5.3.

$$\begin{aligned}
\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t,q,\mu) &\le \mathbf{InSec}^{\text{fil-mac}}_{g}(t,q) + \mathbf{InSec}^{\text{vil-wcr}}_{\mathcal{H}^f}(t,q,\mu) \\
&\le \mathbf{InSec}^{\text{fil-mac}}_{g}(t,q) + \mathbf{InSec}^{\text{fil-wcr}}_{f}\left(2t, \delta(q,\mu)\right) \\
&\le \mathbf{InSec}^{\text{fil-mac}}_{g}(t,q) + \frac{\delta(q,\mu)^2}{2} \cdot \mathbf{InSec}^{\text{fil-mac}}_{f}\left(2t + O(\delta(q,\mu)), \delta(q,\mu)\right)
\end{aligned}$$

$\square$

5.2 MD6 as a Domain Extender for FIL-MACs

It seems somewhat artificial to double the number of bits in the keyspace of MD6 just to prove that it acts as a domain extender for FIL-MACs. Ideally, we would like to prove that it has this property *as is*, while still only making the assumption that the underlying compression functions f and g are FIL-MACs. Unfortunately, at the moment we do not know how to prove this directly (although we have no counterexample for this property). Thus, in order to prove this statement for the standard version of MD6, we will need to introduce an additional assumption on the compression function f.

The idea for the following is that the flag bit z which indicates that the compression function is the final one in the computation of MD6 should "blind" the key values. That is, if we are given oracle access to two functions defined as follows

$$f_0(\cdot) = f(z = 0, K_0, \cdot)$$
$$f_1(\cdot) = f(z = 1, K_1, \cdot)$$

then we should not be able to guess whether $K_0 = K_1$ or not with any significant advantage. The intuition behind this is that we want be able to prove that $\mathcal{M}^f$ is a VIL-MAC with the least assumptions necessary on f to get the proof to go through. Ideally, we would like to be able to only assume that f is a FIL-MAC; however, at the moment we know of no way for doing this. Of course, we could simply assume f is a FIL-PRF and automatically get that $\mathcal{M}^f$ is a VIL-PRF, and hence a VIL-MAC (simply by citing the results of Chapter 4). However, this does not line up with our goal of making minimal assumptions on f. Then a convenient alternative is to assume that f behaves in a "pseudorandom" fashion, but *only* on the z bit of the input.

Definition 25 (Key-Blinding Assumption)**.** Let D be a distinguisher given oracle access to two functions f_0 and f_1 that map $\{0,1\}^{n-1} \to \{0,1\}^c$. We define its advantage as follows.

$$\mathbf{Adv}^{\text{blind}}_{D,f}(t,q) = \left| \Pr\left[\begin{array}{c} K_0 \xleftarrow{\$} \{0,1\}^k; K_1 \xleftarrow{\$} \{0,1\}^k; b \xleftarrow{\$} \{0,1\}; \\ a \leftarrow D^{f(z=0,K_0,\cdot),f(z=1,K_b,\cdot)} \end{array} \ : \ a = b \right] - \frac{1}{2} \right|$$

The goal of D is to try to determine the value of b, i.e. it is trying to tell whether f_0 and f_1 use the same key or not. We can define the overall insecurity of the key-blind property of f to be

$$\mathbf{InSec}_f^{\mathrm{blind}}(t,q) = \max_D \left\{ \mathbf{Adv}_{D,f}^{\mathrm{blind}}(t,q) \right\},$$

where D is given resource constraints of (t,q).

With this assumption, we now prove that the single-keyed version of MD6 acts as a domain extender for FIL-MACs (with the key-blinding assumption). The essential concept here is that by the key-blinding property of f, $\mathcal{M}^f$ behaves almost exactly like $\mathcal{M}^f[2]$ to any algorithm given only oracle access to $\mathcal{M}^f$.

Lemma 5.5. Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$. Then

$$\mathbf{InSec}_{\mathcal{M}^f}^{\text{vil-mac}}(t,q,\mu) \le \mathbf{InSec}_{\mathcal{M}^f[2]}^{\text{vil-mac}}(t,q,\mu) + 2 \cdot \mathbf{InSec}_f^{\mathrm{blind}}(t, \delta(q,\mu)).$$

PROOF. Let A be a forger with the best possible success for attacking the single-keyed version of MD6, $\mathcal{M}^f$. Therefore, given resources t, q, and μ, the probability of A succeeding is $\mathbf{InSec}_{\mathcal{M}^f}^{\text{vil-mac}}(t,q,\mu)$. We use A to construct a distinguisher D that attacks the key-blinding property of f.

Recall that the distinguisher D is given oracle access to two functions $f_0(\cdot) = f(z=0, K_0, \cdot)$ and $f_1(\cdot) = f(z=1, K_1, \cdot)$, which map $\{0,1\}^{n-1} \to \{0,1\}^c$, and must determine whether $K_0 = K_1$.

Algorithm D^{f_0, f_1}

Output: 1 if $K_0 \neq K_1$, 0 otherwise

```
1.  for $i = 1$ to $q$
2.      do $\mathsf{A} \to M_i$
3.          $\mathsf{A} \leftarrow chop_d(f_1(\mathcal{H}^{f_0}(M_i)))$
4.          $\mathsf{A} \to (M, D)$
5.          if $chop_d(f_1(\mathcal{H}^{f_0}(M))) = D$ and for all $i$, $M \neq M_i$
6.              then return 0
7.              else return 1
```

The distinguisher uses A in the following manner: it constructs the function $chop_d \circ$

$f_1 \circ \mathcal{H}^{f_0}$ using its two oracles and responds to hash queries by A with this function. There are two cases:

1. If $b = 1$, then $K_0 \neq K_1$. Therefore the hash function A is querying is
$$chop_d \circ f_1 \circ \mathcal{H}^{f_0} = \mathcal{M}^f[2].$$

2. If $b = 0$, then $K_0 = K_1$. Therefore the hash function A is querying is
$$chop_d \circ f_1 \circ \mathcal{H}^{f_0} = \mathcal{M}^f.$$

Recall that A is an optimal forger for $\mathcal{M}^f$ (although it may also be a forger for $\mathcal{M}^f[2]$ as well), and outputs a forgery (M, D). D outputs 0 if this is a valid forgery (i.e. if A succeeds, then we suspect that $K_0 = K_1$). If the pair is not a valid forgery, we output 1, as we believe that the reason for this failure is that $K_0 \neq K_1$. We can analyze the success probability of D as follows.

$$\begin{aligned}
\Pr[\mathsf{D} \text{ succeeds}] = {} & \Pr[b = 1] \cdot \Pr\left[\mathsf{D}^{f(z=0,K_0,\cdot),\, f(z=1,K_b,\cdot)} \to 1 \mid b = 1\right] \\
& + \Pr[b = 0] \cdot \Pr\left[\mathsf{D}^{f(z=0,K_0,\cdot),\, f(z=1,K_b,\cdot)} \to 0 \mid b = 0\right] \\
= {} & \frac{1}{2} \cdot \Pr\left[\mathsf{A} \text{ fails to forge } \mathcal{M}^f[2]\right] \\
& + \frac{1}{2} \cdot \Pr\left[\mathsf{A} \text{ successfully forges } \mathcal{M}^f\right] \\
\geq {} & \frac{1}{2} \cdot \left(1 - \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t, q, \mu)\right) + \frac{1}{2} \cdot \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t, q, \mu) \qquad (5.1) \\
= {} & \frac{1}{2} + \frac{\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t, q, \mu) - \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t, q, \mu)}{2}
\end{aligned}$$

For Inequality (5.1), note that A could potentially succeed as a forger for $\mathcal{M}^f[2]$. However, its advantage is bounded, as $\mathbf{Adv}^{\text{vil-mac}}_{\mathsf{A},\mathcal{M}^f[2]}(t, q, \mu) \leq \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t, q, \mu)$, whereby we derive the inequality. Therefore,

$$\begin{aligned}\mathbf{Adv}^{\text{blind}}_{D,f}(t,\delta(q,\mu)) &= \left|\Pr[\mathsf{D}\text{ succeeds}] - \frac{1}{2}\right| \\ &\geq \frac{\left|\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t,q,\mu) - \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t,q,\mu)\right|}{2} \\ \mathbf{InSec}^{\text{blind}}_{f}(t,\delta(q,\mu)) &\geq \frac{\left|\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t,q,\mu) - \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t,q,\mu)\right|}{2}\end{aligned}$$

In particular, this implies

$$\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t,q,\mu) \leq \mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f[2]}(t,q,\mu) + 2\cdot\mathbf{InSec}^{\text{blind}}_{f}(t,\delta(q,\mu))$$

□

We now conclude by applying the above lemma to derive a bound on the insecurity of $\mathcal{M}^f$.

Theorem 5.6. Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^c$ be a FIL-MAC and suppose that it has the key-blinding property. Define $g = (\mathit{chop}_d \circ f) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^d$ and suppose that it is also a FIL-MAC. Then $\mathcal{M}^f : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^d$ is a VIL-MAC with

$$\begin{aligned}\mathbf{InSec}^{\text{vil-mac}}_{\mathcal{M}^f}(t,q,\mu) \leq \frac{\delta(q,\mu)^2}{2}\cdot\mathbf{InSec}^{\text{fil-mac}}_{f}(2t+O(\delta(q,\mu)),\delta(q,\mu)) + \\ \mathbf{InSec}^{\text{fil-mac}}_{g}(t,q) + 2\cdot\mathbf{InSec}^{\text{blind}}_{f}(t,\delta(q,\mu)).\end{aligned}$$

Proof. This follows directly by the application of Theorem 5.4 and Lemma 5.5. □

Therefore we have shown that the MD6 mode of operation $\mathcal{M}^f$ acts as a domain extender for FIL-MACs, assuming that both f and g are FIL-MACs and that f has the key-blinding property. Unfortunately we needed to make an additional assumption about the compression function f, and as such this is not a true "domain extension" result. However, at the moment we know of no other way to prove that MD6 has this property, without making additional assumptions.

Chapter 6

Conclusion

In this paper we have demonstrated that the MD6 mode of operation preserves several cryptographic properties of its compression function, namely collision resistance, first-preimage resistance, and pseudorandomness. In addition, we showed that by making a small, but non-trivial, assumption about the key-blinding nature of the compression function, we can demonstrate that MD6 preserves the property of unpredictability as well. We were not able to give a guarantee of property preservation of second-preimage resistance, but we instead reduced to a property with weaker security guarantees.

Future work in this vein might include investigating whether it can be shown that the MD6 mode of operation preserves second-preimage resistance. Also, it seems likely that we ought to be able to show that MD6 preserves unpredictability, without making the additional key-blinding assumption. Dodis et al. [16] demonstrated that a three-keyed version of CBC-MAC preserves unpredictability, but then were able to use clever tricks to reduce it down to a single key (without any additional assumptions). Perhaps similar techniques can be used to show a similar result for MD6. In addition, there are other cryptographic properties, such as Maurer's indifferentiability from a random oracle [23], that seem promising to consider in the context of MD6.

Bibliography

[1] Jee Hea An and Mihir Bellare. Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 1999.

[2] J. Vandewalle B. Preneel, R. Govaerts. Hash Functions Based on Block Ciphers: A Synthetic Approach. In D.R. Stinson, editor, *CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*. Springer, 1994.

[3] M. Bellare, R. Canetti, and H. Krawczyk. The HMAC Construction. *RSA Laboratories CryptoBytes*, 2(1):12–15, 1996.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[5] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security. In *FOCS*, pages 514–523, 1996.

[6] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

[7] Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved Security Analyses for CBC MACs. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 527–545. Springer, 2005.

[8] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[9] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996.

[10] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.

[11] Martin Boesgaard, Thomas Christensen, and Erik Zenner. Badger - A Fast and Provably Secure MAC. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 176–191, 2005.

[12] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. *CoRR*, cs.CR/0010019, 2000. informal publication.

[13] Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998.

[14] Jean-Sbastien Coron, Yevgeniy Dodis, Ccile Malinaud, and Prashant Puniya. Merkle-damgrd revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

[15] Ivan Damgrd. A Design Principle for Hash Functions. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[16] Yevgeniy Dodis, Krzysztof Pietrzak, and Prashant Puniya. A New Mode of Operation for Block Ciphers and Length-Preserving MACs. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 198–219. Springer, 2008.

[17] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[18] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the Cryptographic Applications of Random Functions. In *CRYPTO*, pages 276–288, 1984.

[19] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, 1986.

[20] John Kelsey and Bruce Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2n Work. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005.

[21] Ueli M. Maurer. Indistinguishability of Random Systems. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer, 2002.

[22] Ueli M. Maurer and Krzysztof Pietrzak. Composition of Random Systems: When Two Weak Make One Strong. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 410–427. Springer, 2004.

[23] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[24] Ueli M. Maurer and Johan Sjdin. Single-Key AIL-MACs from Any FIL-MAC. In Lus Caires, Giuseppe F. Italiano, Lus Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 472–484. Springer, 2005.

[25] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 2001.

[26] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[27] Kazuhiko Minematsu and Yukiyasu Tsunoo. Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2006.

[28] NIST. FIPS 180-2 Secure Hash Standard. Available at `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`, August 2002.

[29] NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Available at `http://csrc.nist.gov/groups/ST/hash/index.html`, November 2007.

[30] Krzysztof Pietrzak. *Indistinguishability and Composition of Random Systems.* PhD thesis, ETH Zurich, 2006. Reprint as vol. 6 of *ETH Series in Information Security and Cryptography*, ISBN 3-86626-063-7, Hartung-Gorre Verlag, Konstanz, 2006.

[31] Bart Preneel. The State of Cryptographic Hash Functions. *Lecture Notes in Computer Science*, 1561:158–182, 1999.

[32] Ronald L. Rivest. The MD6 Hash Function. To be released Fall 2008.

[33] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[34] Phillip Rogaway. Formalizing Human Ignorance: Collision-Resistance Hashing without the Keys. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.

[35] Phillip Rogaway and Thomas Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[36] P. Sarkar and P.J. Schellenberg. A Parallelizable Design Principle for Cryptographic Hash Functions. Available at `http://eprint.iacr.org/2002/031`, 2002.

[37] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[38] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

Appendix A

Birthday Paradox

The Birthday Paradox is a famous example of the sometimes counterintuitive nature of combinatorics and probability. It states that in a group of at least 23 randomly chosen people, the probability that any two will share a birthday exceeds 50%.

In general, consider the problem of uniformly sampling q items independently from a set of n items (in the above example, we uniformly sample 23 birthdays from the set of 365 possible birthdays). We wish to provide an upper bound on the probability of picking any item more than once. To generalize further, we consider sampling q elements independently from a set S (for the moment we no longer care about the cardinality of the set S). Instead of drawing from the uniform distribution over S we allow for the distribution on S to be biased. However, this bias is not too great, and we are given an additional parameter p such that the probability of picking any item does not exceed p.

$$\forall x \in S, \ \ \Pr[x] \leq p.$$

Again, our goal in this case is to provide an upper bound on the probability of sampling an identical pair of elements. Parameterized by the number of samples q and the pairwise collision probability p, we call this quantity $P_{coll}(q, p)$.

To illustrate why we choose such a convoluted definition, S might be the output space of the random function $\mathcal{M}^{\mathbf{R}}$, which in this paper is typically $\{0,1\}^d$. In this case, q distinct inputs to $\mathcal{M}^{\mathbf{R}}$ are chosen (perhaps adversarially or not), and the resulting q outputs are our samples. These samples are guaranteed to be independent (over the space of random functions $\mathcal{M}^{\mathbf{R}}$) so long as no bad collisions occur. In addition, for any two samples x and y,

$$\Pr[x = y] \leq \frac{h}{2^c}.$$

Upper bounding $P_{coll}(q,p)$ is fairly simple. If we let E_i be the event that sample x_i is not identical to any of the samples $x_1, x_2, \ldots, x_{i-1}$, then the probability that all E_i are true is equivalent to $1 - P_{coll}(q,p)$ (see Equation (A.1)). We therefore seek to *lower bound* this probability.

$$\begin{aligned}
1 - P_{coll}(q,p) &= \Pr\left[\bigcap_{i=2}^{q} E_i\right] && \text{(A.1)}\\
&= \Pr[E_2]\Pr[E_3 \mid E_2] \cdots \Pr\left[E_q \,\middle|\, \bigcap_{i=2}^{q-1} E_i\right] && \text{(A.2)}\\
&\geq (1-p)(1-2p)\cdots(1-(q-1)p) && \text{(A.3)}\\
&= \prod_{j=2}^{q}(1-(j-1)p)\\
&\geq 1 - p\sum_{j=1}^{q-1} j && \text{(A.4)}\\
&= 1 - p\frac{q(q-1)}{2}
\end{aligned}$$

Equation (A.2) follows from basic conditional probability. For Inequality (A.3), note that $\Pr\left[E_j \,\middle|\, \bigcap_{i=2}^{j-1} E_i\right] \geq 1-(j-1)p$, since if the first $j-1$ samples are distinct then the probability of colliding with any of them cannot exceed $(j-1)p$. Finally, Inequality (A.4) follows from the basic fact that $(1-x)(1-y) \geq (1-x-y)$, for $x, y \geq 0$ (which holds in this case). Therefore,

$$P_{coll}(q,p) \leq \frac{q(q-1)}{2}p.$$