Chapter 5

Hash Functions

A hash function usually means a function that compresses, meaning the output is shorter than the input. Often, such a function takes an input of arbitrary or almost arbitrary length to one whose length is a fixed number, like 160 bits. Hash functions are used in many parts of cryptography, and there are many different types of hash functions, with differing security properties. We will consider them in this chapter.

5.1 The hash function SHA1

The hash function known as SHA1 is a simple but strange function from strings of arbitrary length to strings of 160 bits. The function was finalized in 1995, when a FIPS (Federal Information Processing Standard) came out from the US National Institute of Standards that specified SHA1.

The function SHA1: $\{0,1\}^* \to \{0,1\}^{160}$ is conveniently described in terms of a lower-level function, sha1: $\{0,1\}^{512} \times \{0,1\}^{160} \to \{0,1\}^{160}$ called its *compression function*. Let's begin by defining the sha1 compression function, and then we'll define SHA1 from it.

First we need some notation and definitions. If A and B are 32-bit strings then we let $A+B$ denote the 32-bit string that you get by treating A and B as nonnegative numbers, adding them modulo 2^{32}, and treating the result as a 32-bit strings. This is just the usual notion of "computer addition." If $A = A_0A_1 \ldots A_{31}$ is a 32-bit string (each A_i a bit) and $i \in [0\,..\,32]$ is a number, then we let $A \lll i$ be the circular left shift of $A = A_0A_1 \ldots A_{31}$ by i positions, meaning $A_{i \bmod 32}A_{i+1 \bmod 32} \ldots A_{i+31 \bmod 32}$.

For $0 \le t \le 19$, set $K_t =$ 0x5a827999 and $f_t(B,C,D) = (B \wedge C) \vee (\overline{B} \wedge D)$. For $20 \le t \le 39$ set $K_t =$ 0x6ed9eba1 and $f_t(B,C,D) = B \oplus C \oplus D$. For $40 \le t \le 59$ set $K_t =$ 0x8f1bbcdc and $f_t(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$. For $60 \le t \le 79$ set $K_t =$ 0xca62c1d6 and $f_t(B,C,D) = B \oplus C \oplus D$. Now the SHA1 compression function is defined in Figure 5.1.

algorithm sha1(X, H) // where $|X| = 512$ and $|H| = 160$
10 Parse X into $W_1 \cdots W_{16}$ where $|W_1| = \cdots = |W_{16}| = 32$
11 Parse H into $H_0H_1H_2H_3H_4$ where $|H_0| = |H_1| = |H_2| = |H_3| = |H_4| = 32$
12 **for** $t \leftarrow 16$ **to** 79 **do** $W_t \leftarrow (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$
13 $A \leftarrow H_0$, $B \leftarrow H_1$, $C \leftarrow H_2$, $D \leftarrow H_3$, $E \leftarrow H_4$
14 **for** $t \leftarrow 0$ **to** 79 **do**
15 $T \leftarrow A \lll 5 + f_t(B, C, D) + E + W_t + K_t$
16 $E \leftarrow D$, $D \leftarrow C$, $C \leftarrow B \lll 30$, $B \leftarrow A$, $A \leftarrow T$
17 $H_0 \leftarrow H_0 + A$, $H_1 \leftarrow H_1 + B$, $H_2 \leftarrow H_2 + C$, $H_3 \leftarrow H_3 + D$, $H_4 \leftarrow H_4 + E$
18 **return** $H_0H_1H_2H_3H_4$

Figure 5.1: The function sha1, the compression function of SHA1.

algorithm SHA1(M)
20 $M^* \leftarrow M \parallel 10^i \parallel [|M|]_{64}$ where $i \geq 0$ is the smallest such that $|M| + 1 + i + 64$ is divisible by 512
21 Partition M^* into $M_1 \ldots M_m$
22 $Y_0 \leftarrow$ 0x67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0
23 **for** $i \leftarrow 1$ **to** m **do**
24 $Y_i \leftarrow \text{SHA}(M_i, Y_{i-1})$
25 **return** Y_m

Figure 5.2: The function SHA1.

To build SHA1 out of sha1 is easy. The method is defined in Figure 5.2. Recall that when $\ell \geq 0$ is a number, $[\ell]_{64}$ is the 64-bit string that is the binary encoding of ℓ mod 2^{64}.

Now that we have defined SHA1, it is natural to ask where such a strange function comes from—why is it the way that it is? Such questions don't really have answers. SHA1 is derived from a function called MD4 that was proposed by Ron Rivest in 1990, and the key ideas behind SHA1 are already in MD4. Besides SHA1, another well-known "child" of MD4 is MD5, which was likewise proposed by Rivest. The MD4, MD5, and SHA1 are all quite similar in structure. The first two produce a 128-bit output, and work by "chaining" a compression function that goes from $512 + 128$ bits to 128 bits, while SHA1 works by chaining a compression function from $512 + 160$ bits to 160 bits.

So what is SHA1 supposed to do? First and foremost, it is supposed to be the case that nobody can find distinct strings M and M' such that $\text{SHA1}(M) = \text{SHA1}(M')$. This property is called *collision resistance*.

Stop for a moment and think about the collision-resistance requirement, for it is really quite amazing to think that such a thing could be possible. The function SHA1 maps strings of any length to strings of 160 bits. So even if you restricted the

domain of SHA1 just to "short" strings—let us say strings of length 256 bits—then there must be an *enormous* number of pairs of strings M and M' that hash to the same value. This is just by the pigeonhole principle: if 2^{256} pigeons (the 256-bit messages) roost in 2^{160} holes (the 160-bit hash values) then some two pigeons (two distinct strings) roost in the same hole (have the same hash). Indeed countless pigeons must share the same hole. The difficult is only that nobody knows how to *identify* even two such pigeons!

In trying to define this collision-resistance property of SHA1 we immediately run into "foundational" problems. We would like to say that it is computationally infeasible to output a pair of distinct strings M and M' that collide under SHA1. But in what sense could it be infeasible? There *is* a program—indeed a very short an simple one, having just two "print" statements—whose output specifies a collision. It's not computationally hard to output a collision; it can't be. The only difficulty is our *human* problem of not knowing what this program is.

Mathematics doesn't care about what us humans know and it has no way to talk about such things. It would thus seem to be impossible to make a mathematical definition that captures the idea that human beings can't find collisions in SHA1. In order to reach a mathematically precise definition we are going to have to change the very nature of what we conceive to be a hash function. This is unfortunate, in some way, because it distances us from concrete hash functions like SHA1. But no alternative is know.

5.2 Collision-resistant hash functions

Although one often thinks of a hash function as a single function, in our formulation, a *hash function* is a family of functions $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$. Later we will see why it is important to consider families rather than merely consider single functions.

Here is some notation we use in this chapter. Let $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function. For any key K and $y \in \mathsf{Range}(H)$ we let

$$H_K^{-1}(y) \;=\; \{\, x \in \mathsf{Dom}(H) \;:\; H_K(x) = y \,\}$$

denote the pre-image set of y under H_K. Let

$$\mathsf{Image}(H_K) \;=\; \{\, H_K(x) \;:\; x \in \mathsf{Dom}(H) \,\}$$

denote the image of H_K.

Let us begin with the most popular class, namely collision-resistant hash functions. Let $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function. A *collision* for an instance H_K of H is a pair $x_1, x_2 \in \mathsf{Dom}(H)$ of distinct points in the domain such that $H_K(x_1) = H_K(x_2)$. The most basic security property of a hash function is collision-resistance, which measures the ability of an adversary to find a collision for H_K. There are different notions of collision-resistance, varying in restrictions put on the adversary in its quest for a collision.

Pre-key attack phase	A selects $2 - s$ points
Key selection phase	A key K is selected at random from $\mathsf{Keys}(H)$
Post-key attack phase	A is given K and returns s points
Winning condition	The 2 points selected by A form a collision for H_K

Figure 5.3: Framework for security notions for collision-resistant hash functions. The three choices of $s \in \{0, 1, 2\}$ give rise to three notions of security.

To introduce the different notions, we imagine a game, parameterized by an integer $s \in \{0, 1, 2\}$, and involving an adversary A. It consists of a *pre-key attack* phase, followed by a *key-selection phase*, followed by a *post-key attack phase*. The adversary is attempting to find a collision for H_K, where key K is selected at random from $\mathsf{Keys}(H)$ in the key-selection phase. Recall that a collision consists of a pair x_1, x_2 of (distinct) points in $\mathsf{Dom}(H)$. The adversary is required to specify $2 - s$ points in the pre-key attack phase, before it has any information about the key. (The latter has yet to be selected.) Once the adversary has specified these points and the key has been selected, the adversary is given the key, and will choose the remaining s points as a function of the key, in the post-key attack phase. It wins if the $2 = (2 - s) + s$ points it has selected form a collision for H_K.

Figure 5.3 summarizes the framework. The three choices of the parameter s give rise to three notions of security. The higher the value of s the more power the adversary has, and hence the more stringent is the corresponding notion of security. Figure 5.4 provides in more detail the experiments underlying the three attacks arising from the above framework. We represent by st information that the adversary wishes to maintain across its attack phases. It will output this information in the pre-key attack phase, and be provided it at the start of the post-key attack phase.

In a variant of this model that we consider in Section 5.7, the adversary is not given the key K in the post-key attack phase, but instead is given an oracle for $H_K(\cdot)$. To disambiguate, we refer to our current notions as capturing collision-resistance under *known-key* attack, and the notions of Section 5.7 as capturing collision-resistance under *hidden-key* attack. The notation in the experiments of Figure 5.4 and Definition 5.1 reflects this via the use of "kk", except that for CR0, known and hidden key attacks coincide, and hence we just say cr0.

The three types of hash functions we are considering are known by other names in the literature, as indicated in Figure 5.5.

Definition 5.1 Let H: $\mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function and let A be an algorithm. We let

$$\mathbf{Adv}^{\text{cr2-kk}}_{H}(A) \;=\; \Pr\left[\mathbf{Exmt}^{\text{cr2-kk}}_{H}(A) = 1\right]$$

$\mathbf{Exmt}_H^{\text{cr2-kk}}(A)$ $K \xleftarrow{\$} \mathsf{Keys}(H)$; $(x_1, x_2) \xleftarrow{\$} A(K)$ If $x_1 \neq x_2$ and $H_K(x_1) = H_K(x_2)$ then return 1 else return 0
$\mathbf{Exmt}_H^{\text{cr1-kk}}(A)$ $(x_1, st) \xleftarrow{\$} A()$; $K \xleftarrow{\$} \mathsf{Keys}(H)$; $x_2 \xleftarrow{\$} A(K, st)$ If $x_1 \neq x_2$ and $H_K(x_1) = H_K(x_2)$ then return 1 else return 0
$\mathbf{Exmt}_H^{\text{cr0}}(A)$ $(x_1, x_2) \xleftarrow{\$} A()$; $K \xleftarrow{\$} \mathsf{Keys}(H)$ If $x_1 \neq x_2$ and $H_K(x_1) = H_K(x_2)$ then return 1 else return 0

Figure 5.4: Experiments defining security notions for three kinds of collision-resistant hash functions under known-key attack.

$$\begin{aligned} \mathbf{Adv}_H^{\text{cr1-kk}}(A) &= \Pr\left[\mathbf{Exmt}_H^{\text{cr1-kk}}(A) = 1\right] \\ \mathbf{Adv}_H^{\text{cr0}}(A) &= \Pr\left[\mathbf{Exmt}_H^{\text{cr0}}(A) = 1\right] . \end{aligned}$$

For any t we define

$$\begin{aligned} \mathbf{Adv}_H^{\text{cr2-kk}}(t, m) &= \max_A\{\mathbf{Adv}_H^{\text{cr2-kk}}(A)\} \\ \mathbf{Adv}_H^{\text{cr1-kk}}(t, m) &= \max_A\{\mathbf{Adv}_H^{\text{cr1-kk}}(A)\} \end{aligned}$$

where the maximum is over all A having time-complexity t and where the sum of the lengths of the two messages in the collision found by A is at most m bits. Finally, we let

$$\mathbf{Adv}_H^{\text{cr0}} = \max_A\{\mathbf{Adv}_H^{\text{cr0}}(A)\}$$

where the maximum is over all A, regardless of A's time-complexity. All these definitions assume adversaries are *legitimate* in the sense that the points that they output are in the domain of the hash function. ▮

We use our usual conventions. Time-complexity refers to the worst-case running time of the entire experiment, including the running time of the adversary, the size of its code, and the time for the various hash function computations in the experiment. Although there is formally no definition of a "secure" hash function, we will talk of a hash function being CR2, CR1 or CR0 with the intended meaning that its associated advantage function is small for practical values of the time-complexity.

Type	Name(s) in literature
CR2-KK	collision-free, collision-resistant, collision-intractable
CR1-KK	universal one-way [29] (aka. target-collision resistant [**?**])
CR0	universal, almost universal

Figure 5.5: Types of hash functions, with names in our framework and corresponding names found in the literature.

There is no time parameter in the CR0 case since the optimal adversary has very low time-complexity. It simply has hardwired into its code a "best" choice of distinct points x_1, x_2, meaning a choice for which

$$\begin{aligned} &\Pr\left[\, K \xleftarrow{\$} \mathsf{Keys}(H) \;:\; H_K(x_1) = H_K(x_2) \,\right] \\ &= \max_{y_1 \neq y_2} \Pr\left[\, K \xleftarrow{\$} \mathsf{Keys}(H) \;:\; H_K(y_1) = H_K(y_2) \,\right] . \end{aligned}$$

Note this value equals $\mathbf{Adv}^{\mathrm{cr0}}_H$.

Clearly, a CR2 hash function is also CR1 and a CR1 hash function is also CR0. The following states the corresponding relations formally and quantitatively. The proof is trivial and is omitted. Here t is assumed large enough that the trivial CR0 adversary discussed above has time-complexity bounded above by t.

Proposition 5.2 Let $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function. Then

$$\mathbf{Adv}^{\mathrm{cr0}}_H \;\leq\; \mathbf{Adv}^{\mathrm{cr1\text{-}kk}}_H(t,m) \;\leq\; \mathbf{Adv}^{\mathrm{cr2\text{-}kk}}_H(t,m) \,.$$

∎

5.3 One-wayness of collision-resistant hash functions

We briefly considered the notion of one-wayness of a function h in the context of the Unix password hashing scheme in Section 3.10. Here we consider the more general notion of one-wayness of a family of functions H and its relation to collision-resistance.

Intuitively, a family H is one-way if it is computationally infeasible, given H_K and a range point $y = K_K(x)$, where x was chosen at random from the domain, to find a pre-image of y (whether x or some other) under H_K. (Definition 3.18 can be viewed as the special case that the family contains exactly one function, meaning the key-space has size one.) Since this definition too has a hidden-key version, we indicate the known-key in the notation below.

Definition 5.3 Let $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a family of functions and let A be an algorithm. We consider the following experiment:

$\mathbf{Exmt}^{\text{ow-kk}}_H(A)$
$K \xleftarrow{\$} \mathsf{Keys}(H)$; $x \xleftarrow{\$} \mathsf{Dom}(H)$; $y \leftarrow H_K(x)$; $x' \xleftarrow{\$} A(K, y)$
If $H_K(x') = y$ then return 1 else return 0

We let

$$\mathbf{Adv}^{\text{ow-kk}}_H(A) \;=\; \Pr\left[\mathbf{Exmt}^{\text{ow-kk}}_H(A) = 1\right] .$$

For any t we define

$$\mathbf{Adv}^{\text{ow-kk}}_H(t) \;=\; \max_A\{\mathbf{Adv}^{\text{ow-kk}}_H(A)\}$$

where the maximum is over all A having time-complexity t. This definition assumes adversaries are *legitimate* in the sense that the point x' that they output is in the domain of the hash function. ▌

We now ask ourselves whether collision-resistance implies one-wayness. It is easy to see, however, that, in the absence of additional assumptions about the hash function than collision-resistance, the answer is "no." For example, let H be a family of functions every instance of which is the identity function. Then H is highly collision-resistant (the advantage of an adversary in finding a collision is zero regardless of its time-complexity since collisions simply don't exist) but is not one-way.

However, we would expect that "genuine" hash functions, meaning ones that perform some non-trivial compression of their data (ie. the size of the range is more than the size of the domain) are one-way. This turns out to be true, but needs to be carefully quantified. To understand the issues, it may help to begin by considering the natural argument one would attempt to use to show that collision-resistance implies one-wayness.

Suppose we have an adversary A that has a significant advantage in attacking the one-wayness of hash function H. We could try to use A to find a collision via the following strategy. In the pre-key phase (we consider a type-1 attack) we pick and return a random point x_1 from $\mathsf{Dom}(H)$. In the post-key phase, having received the key K, we compute $y = H_K(x_1)$ and give K, y to A. The latter returns some x_2, and, if it was successful, we know that $H_K(x_2) = y$. So $H_K(x_2) = H_K(x_1)$ and we have a collision.

Not quite. The catch is that we only have a collision if $x_2 \neq x_1$. The probability that this happens turns out to depend on the quantity:

$$\mathbf{PreIm}_H(1) \;=\; \Pr\left[\, K \xleftarrow{\$} \mathsf{Keys}(H)\,;\, x \xleftarrow{\$} \mathsf{Dom}(H)\,;\, y \leftarrow H_K(x) \;:\; |H_K^{-1}(y)| = 1 \,\right] .$$

This is the probability that the size of the pre-image set of y is exactly 1, taken over y generated as shown. The following Proposition says that a collision-resistant function H is one-way as long as $\mathbf{PreIm}_H(1)$ is small.

Proposition 5.4 Let H: $\mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function, and assume $\mathsf{Dom}(H)$ contains strings of length at most m. Then

$$\mathbf{Adv}^{\text{ow-kk}}_{H}(t) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{cr1-kk}}_{H}(t, m) + \mathbf{PreIm}_{H}(1) \;.$$

∎

The result is about the CR1 type of collision-resistance. However Proposition 5.2 implies that the same is true for CR2.

A general and widely-applicable corollary of the above Proposition is that collision-resistance implies one-wayness as long as the domain of the hash function is significantly larger than its range. The following quantifies this.

Corollary 5.5 Let H: $\mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function, and assume $\mathsf{Dom}(H)$ contains strings of length at most m. Then

$$\mathbf{Adv}^{\text{ow-kk}}_{H}(t) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{cr1-kk}}_{H}(t, m) + \frac{|\mathsf{Range}(H)|}{|\mathsf{Dom}(H)|} \;.$$

∎

In particular, we will later see hash functions like SHA-1 where the range consists of 160 bit strings but the domain consists of strings of length up to 2^{512} bits. We would apply the above by letting $\mathsf{Dom}(H)$ be the $\{0,1\}^{\ell}$ for ℓ and letting H be the family containing just one function, this being the restriction of SHA-1 to inputs of length ℓ. The ratio of range size to domain size is $2^{160-\ell}$ thus we see that collision-resistance implies one-wayness for such functions, with the added term in bound of Corollary 5.5 getting smaller as ℓ increases.

There are some natural hash functions, however, for which Corollary 5.5 does not apply. Consider a hash function H every instance of which is two-to-one. The ratio of range size to domain size is 1/2, so the right hand side of the equation of Corollary 5.5 is 1, meaning the bound is vacuous. However, such a function is a special case of the one considered in the following example, and by direct use of Proposition 5.4 one can show that collision-resistance does in fact imply one-wayness.

Example 5.6 Suppose $1 \leq r < d$ and let H: $\mathsf{Keys}(H) \times \{0,1\}^{d} \to \{0,1\}^{r}$ be a hash function which is *regular*, meaning that for every key K, all points in the image of H_K have the same pre-image size. In other words, $|H_K^{-1}(y)| = 2^{d-r}$ for every $y \in \mathsf{Image}(H_K)$. (Above we discussed the case $d = r + 1$.) The assumption $d > r$ then implies that $\mathbf{PreIm}_{H}(1) = 0$. So Proposition 5.4 tells us that

$$\mathbf{Adv}^{\text{ow-kk}}_{H}(t) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{cr1-kk}}_{H}(t, d) \;.$$

In other words, if H is collision-resistant then it is a OWF with hardly any loss of security. ∎

We now turn to proving the above claims, beginning with Corollary 5.5 and then moving to Proposition 5.4.

Proof of Corollary 5.5: For any key K, the number of points in the range of H_K that have exactly one pre-image certainly cannot exceed $|\mathsf{Range}(H)|$. This implies that

$$\mathbf{PreIm}_H(1) \;\leq\; \frac{|\mathsf{Range}(H)|}{|\mathsf{Dom}(H)|} \;.$$

The corollary follows from Proposition 5.4. ▮

Proof of Proposition 5.4: We associate to a given adversary B an adversary A such that

$$\mathbf{Adv}^{\text{ow-kk}}_H(B) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{cr1-kk}}_H(A) + \mathbf{PreIm}_H(1) \tag{5.1}$$

and furthermore the time-complexity of A is that of B. The Proposition follows. Here's how A works:

Pre-key phase	Post-key phase
Adversary $A()$	Adversary $A(K, st)$
$\quad x_1 \xleftarrow{\$} \mathsf{Dom}(H)$; $st \leftarrow x_1$	$\quad$ Retrieve x_1 from st
$\quad$ Return (x_1, st)	$\quad y \leftarrow H_K(x_1)$; $x_2 \xleftarrow{\$} B(K, y)$
	$\quad$ Return x_2

Consider the experiment

$$x_1 \xleftarrow{\$} \mathsf{Dom}(H)\,;\; K \xleftarrow{\$} \mathsf{Keys}(H)\,;\; y \leftarrow H_K(x_1)\,;\; x_2 \xleftarrow{\$} B(K, y)\;.$$

Let $\Pr[\cdot]$ denote the probability of event "$\cdot$" in this experiment. Define the following events:

$$\begin{aligned} C &: H_K(x_1) = H_K(x_2) \\ D &: x_1 = x_2 \\ G &: |H_K^{-1}(y)| \geq 2\;. \end{aligned}$$

Then

$$\begin{aligned} \mathbf{Adv}^{\text{ow-kk}}_H(B) &= \Pr[C] \\ &= \Pr[C \wedge D \wedge G] + \Pr\left[C \wedge \overline{D} \wedge G\right] + \Pr\left[C \wedge \overline{G}\right] \\ &\leq \Pr[C \wedge D \wedge G] + \Pr\left[C \wedge \overline{D} \wedge G\right] + \Pr\left[\overline{G}\right] \\ &= \Pr[C \wedge D \wedge G] + \Pr\left[C \wedge \overline{D} \wedge G\right] + \mathbf{PreIm}_H(1) \end{aligned}$$

If G and C are true then the probability of D cannot be more than the probability of $\overline{D}$ because there are at least two points in the set $H_K^{-1}(y)$. Thus

$$\Pr[C \wedge D \wedge G] \;=\; \Pr[D \mid C \wedge G] \cdot \Pr[C \wedge G]$$

$$\begin{aligned} &\leq \Pr\left[\overline{D} \mid C \wedge G\right] \cdot \Pr\left[C \wedge G\right] \\ &= \Pr\left[C \wedge \overline{D} \wedge G\right] . \end{aligned}$$

From the above we now get

$$\begin{aligned} \mathbf{Adv}^{\text{ow-kk}}_{H}(B) &\leq 2 \cdot \Pr\left[C \wedge \overline{D} \wedge G\right] + \mathbf{PreIm}_H(1) \\ &= 2 \cdot \mathbf{Adv}^{\text{cr1-kk}}_{H}(A) + \mathbf{PreIm}_H(1) . \end{aligned}$$

This yields Equation (5.1) and concludes the proof. ▌

5.4 The MD transform

Let Keys be a finite set and let $b, c \geq 1$ be integers called, respectively, the block-length and the chaining-length. Let $H\colon \mathsf{Keys} \times \{0,1\}^{b+c} \to \{0,1\}^c$ be a given hash function that we will call the *compression function*, having, as the notation indicates, key-space Keys, domain $\{0,1\}^{b+c}$ and range $\{0,1\}^c$. Our goal is to build a hash function $\overline{H}$ over the larger domain

$$D_b \;=\; \{\, M[1]\ldots M[n] \;:\; 1 \leq n < 2^b \text{ and } M[i] \in \{0,1\}^b \text{ for } 1 \leq i \leq n \,\} .$$

The constructed hash function $\overline{H}\colon \mathsf{Keys} \times D_b \to \{0,1\}^c$ has the same set of keys and range as H. To define it, we fix an element $\mathrm{IV} \in \{0,1\}^c$, for example $\mathrm{IV} = 0^c$. We recall that $[i]_b$ denotes the b bit binary representation of the integer $i \in \{0,\ldots,2^b - 1\}$. (We stress that the length of the string $[i]_b$ is always exactly b bits: left-padding with zeros is used if the binary representation of i is strictly less than b-bits long.) Given a key K and an input $M = M[1]\ldots M[n] \in D_b$, we define

$\overline{H}_K(M[1]\ldots M[n])$
  $C[0] \leftarrow \mathrm{IV}\ ;\ M[n+1] \leftarrow [n]_b$
  For $i = 1,\ldots,n+1$ do $C[i] \leftarrow H_K(M[i] \,\|\, C[i-1])$ EndFor
  Return $C[n+1]$

We refer to this process of turning H into $\overline{H}$ as the *MD transform*.

The value of the MD transform is that it preserves CR2. In other words, if the compression function H is CR2 then so is the hash function $\overline{H}$. The following theorem states this more formally.

Theorem 5.7 Let $H\colon \mathsf{Keys} \times \{0,1\}^{b+c} \to \{0,1\}^c$ be a compression function and $\overline{H}\colon \mathsf{Keys} \times D_b \to \{0,1\}^c$ the hash function resulting from the MD transform as above. Then

$$\mathbf{Adv}^{\text{cr2-kk}}_{\overline{H}}(t,m) \;\leq\; \mathbf{Adv}^{\text{cr2-kk}}_{H}(t', b+c)$$

where $t' = t + O(m)$. ▌

Proof of Theorem 5.7: Let $\overline{A}$ be an adversary attacking $\overline{H}$. We associate to it an adversary A attacking H such that

$$\mathbf{Adv}^{\text{cr2-kk}}_{\overline{H}}(\overline{A}) \;\leq\; \mathbf{Adv}^{\text{cr2-kk}}_{H}(A)$$

and furthermore the time-complexity of A is that of $\overline{A}$ plus an amount proportional to the length of the two messages output by A as its collision. This proves the theorem. We now proceed to construct and analyze A. It works as follows:

Adversary $A(K)$
  $(M_1, M_2) \xleftarrow{\$} \overline{A}(K)$
  Break M_1 into b-bit blocks as $M_1[1] \ldots M_1[n_1]$
  Break M_2 into b-bit blocks as $M_2[1] \ldots M_2[n_2]$
  $M_1[n_1+1] \leftarrow [n_1]_b$; $M_2[n_2+1] \leftarrow [n_2]_b$
  $C_1[0] \leftarrow \text{IV}$; $C_2[0] \leftarrow \text{IV}$
  For $i = 1, \ldots, n_1 + 1$ do $C_1[i] \leftarrow H_K(M_1[i] \,\|\, C_1[i-1])$ EndFor
  For $i = 1, \ldots, n_2 + 1$ do $C_2[i] \leftarrow H_K(M_2[i] \,\|\, C_2[i-1])$ EndFor
  If $n_1 \neq n_2$ then return $(M_1[n_1+1] \,\|\, C_1[n_1], M_2[n_2+1] \,\|\, C_2[n_2])$
  Else
    For $i = n_1, \ldots, 1$ do
      If $M_1[i] \,\|\, C_1[i-1] \neq M_2[i] \,\|\, C_2[i-1]$
        then return $(M_1[i] \,\|\, C_1[i-1], M_2[i] \,\|\, C_2[i-1])$
    EndFor
  EndIf

∎

5.5 Polynomial evaluation is an almost-universal hash function

5.6 The CBC MAC is an almost-universal hash function

We show that if $M, M' \in (\{0,1\}^n)^+$ are distinct then $\Pr[\rho \xleftarrow{\$} \mathsf{Rand}(n) : \text{CBC}_\rho(M) = \text{CBC}_\rho(M')]$ is small. By "small" we mean a slowly growing function of $m = \|M\|_n$ and $m' = \|M'\|_n$. We use another game-playing argument.

Lemma 5.8 [CBC Collision Bound] Fix $n, m, m' \geq 1$ and let $M \in (\{0,1\}^n)^m$ and $M' \in (\{0,1\}^n)^{m'}$ be distinct strings. Then

$$\mathbf{Coll}^{m,\,m'}_{\text{CBC}[\mathsf{Rand}(n)]} \;\leq\; \frac{mm'}{2^n} + \frac{\max\{m, m'\}}{2^n}$$

∎

Proof: If the lengths of M and M' differ then, without loss of generality, let M name the shorter of the two strings and let M' name the longer. Then write $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_{m'}$ where each M_i and each M'_j is n-bits long and $m' \geq m$. Write M as $M = N \,\|\, I$ and write M' as $M' = N \,\|\, I'$ where N is the longest common prefix of M and M' whose length is divisible by n. Let $k = \|N\|_n$. Note that $k = m$ if M is a prefix of M' and otherwise k is the largest nonnegative integer such that $M_1 \cdots M_k = M'_1 \cdots M'_k$ but $M_{k+1} \neq M'_{k+1}$. By definition, if $M_1 \neq M'_1$ then $k = 0$. Further note that $k < m'$.

Consider game C1, as defined in Figure 5.6. This game realizes one way to compute $Y_m = \mathrm{CBC}_\rho(M)$ and $Y'_{m'} = \mathrm{CBC}_\rho(M')$ for a random $\rho \in \mathsf{Rand}(n)$, and so we would like to bound the probability, in game C1, that $Y_m = Y'_{m'}$. Also depicted in Figure 5.6 is game C2, obtained by eliminating the shaded statements.

Before taking up the analysis of the games in earnest, we give a bit of intuition about what they aim to capture. We are interested in the probability that $Y_m = Y'_{m'}$. This can happen due either to an *internal collision* between X_m and $X'_{m'}$, where Y_m was produced as $\rho(X_m)$ and $Y'_{m'}$ was produced as $\rho(X'_{m'})$, or Y_m might equal $Y'_{m'}$ even in the absence of this collsion between X_m and $X'_{m'}$. Intuitively, the latter seems unlikely, and it is. The former is unlikely too, but to prove this we generalize and consider a broader set of internal collisions than just X_m coinciding with $X'_{m'}$. That is, consider the $X_1, \ldots, X_m$ values that get fed to ρ as we process $M = M_1 \cdots M_m$, and consider the $X'_1, \ldots, X'_{m'}$ values that get fed to ρ as we process $M' = M'_1 \cdots M'_{m'}$. Accounting for the fact that $X_i = X'_i$ for all $i \in [1 \,..\, k]$ (that is, the X_i-values that occur as we process the common prefix N of M and M'), we don't really expect to see collisions among $X_1, \ldots, X_m, X'_{i+1}, \ldots, X'_{m'}$. To get a better bound, we will "give up" in the analysis for some of these possibilities, but not all. Specifically, the internal collisions that we focus on are (a) collisions that occur while we process N (meaning collisions among any of $X_1, \ldots, X_k$); (b) collisions that occur between N and I (those between one of $X_1, \ldots, X_k$ and one of $X_{k+1}, \ldots, X_m$); (c) collisions that occur between N and I' (one of $X_1, \ldots, X_k$ coincides with one of $X'_{k+1}, \ldots, X'_{m'}$); and (d) collisions that occur between I' and I (one of $X_{k+1}, \ldots, X_m$ coincides with one of $X'_{k+1}, \ldots, X'_{m'}$). There is nothing "magical" about giving up exactly on these internal collisions—it is simply that the more internal collisions one gives up on the worse the bound but the easier the analysis. The proof we give formalizes the intuition of this paragraph and does all the necessary accounting.

Now look at game C1. It works not by growing a single random function ρ but by growing *three* random functions: η, ι, and ι'. The function η keeps track of the association of points that arise during the processing of N, the function ι keeps track of the association of points that arise during the processing of I, and the function ι' keeps track of the association of points that arise during the processing of I'. If a value X should get placed in the domain of two or more of these three functions then we regard the corresponding range value as that specified first by η if such a value has been specified, and as the value secondarily by ι otherwise. Game C1 can

```
01  bad ← false
02  for X ∈ {0,1}^n do η(X) ← ι(X) ← ι'(X) ← undef

10  Y_0 ← 0^n
11  for i ← 1 to k do
12      Y_i ←$ {0,1}^n
13      X_i ← Y_{i−1} ⊕ M_i
14      if X_i ∈ Domain(η) then Y_i ← η(X_i), bad ← true else η(X_i) ← Y_i

20  for i ← k+1 to m do
21      Y_i ←$ {0,1}^n
22      X_i ← Y_{i−1} ⊕ M_i
23      if X_i ∈ Domain(ι) then Y_i ← ι(X_i) else ι(X_i) ← Y_i
24      if X_i ∈ Domain(η) then Y_i ← η(X_i), bad ← true

30  Y'_k ← Y_k
40  for j ← k+1 to m' do
41      Y'_j ←$ {0,1}^n
42      if Y'_j = Y_m then bad ← true
43      X'_j ← Y'_{j−1} ⊕ M'_j
44      if X'_j ∈ Domain(ι') then Y_j ← ι'(X'_j) else ι'(X'_j) ← Y'_j
45      if X'_j ∈ Domain(ι) then Y'_j ← ι(X'_j), bad ← true
46      if X'_j ∈ Domain(η) then Y'_j ← η(X'_j), bad ← true
```

Figure 5.6: Game C1, as written, and game C2, after eliminating the shaded statements. Game C1 provides a way to compute the CBC MAC of distinct messages $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_{m'}$ that are identical up to block k. The computed MACs are Y_m and $Y'_{m'}$.

thus be seen to provide a perfect simulation of the CBC algorithm over a random function $\rho \in \mathsf{Rand}(n)$, and we seek to bound the probability, in game C1, that $Y_m = Y'_{m'}$.

We first claim that, in game C1, any time that $Y_m = Y'_{m'}$ it is also the case that flag *bad* gets set to true. Focus on line 42. Since $m' \geq k+1$ we know that the loop covering lines 40–46 will be executed at least once and $Y'_{m'}$ will take its value as a result of these statements. When a preliminary $Y'_{m'}$ value gets chosen at line 41 for iteration $j = m'$ we see that if $Y'_{m'} = Y_m$ then line 42 will set the flag *bad*. But observe that the initially chosen $Y'_{m'}$ value is not necessarily the final $Y'_{m'}$ value returned by game C1—the $Y'_{m'}$ value chosen at line 42 could get overwritten at any of lines line 44, 45, or 46. If the value gets overwritten at either of line 45 or line 46

then *bad* will get set to true. If $Y'_{m'}$ gets overwritten by Y_m at line 44 then Y_m had earlier been placed in the range of ι' and it could only have gotten there (since ι' grows only by the **else** -clause of line 44) by $Y'_{m'}$ being equal to Y'_j for an already selected $j \in [k+1 \,..\, m'-1]$. In that case the flag *bad* would have already been set to true by an earlier execution of line 42: when the Y'_j value was randomly selected at line 41 and found to coincide with Y_m the flag *bad* is set. We thus have that every execution of game C1 that results in $Y_m = Y'_{m'}$ also results in *bad* being true.

Let $\Pr_1[\cdot]$ denote the probability of an event in game C1 and let $\Pr_2[\cdot]$ denote the probability of an event in game C2. Let B be the event (whether in game C1 or game C2) that the flag *bad* gets set to true. By the contents of the last paragraph, $\Pr_\rho[\mathrm{CBC}_\rho(M) = \mathrm{CBC}_\rho(M')] \leq \Pr_1[\mathsf{B}]$. Also note that games C1 and C2 are identical until the flag *bad* gets set to true (meaning that any execution in which a highlighted statement is executed it is also the case that *bad* is set to true in that execution) and so, in particular, $\Pr_1[\mathsf{B}] = \Pr_2[\mathsf{B}]$. We thus have that $\Pr_\rho[\mathrm{CBC}_\rho(M) = \mathrm{CBC}_\rho(M')] \leq \Pr_2[\mathsf{B}]$. We now proceed to bound $\Pr_2[\mathsf{B}]$, showing that $\Pr_2[\mathsf{B}] \leq mm'/2^n + \max\{m, m'\}/2^n$. This is done by summing the probabilities that *bad* gets set to true at lines 14, 24, 42, 45, and 46.

The probability that *bad* gets set at **line 14** of game C2 is at most $0.5\, k(k-1)/2^n$. This is because every point placed into the domain of η is of the form $X_i = Y_{i-1} \oplus M_i$ where each Y_{i-1} is randomly selected from $\{0,1\}^n$ (at line 12) with the single exception of Y_0, which is a constant. So for any i and j with $1 \leq i < j \leq k$ we have that $\Pr[Y_{i-1} \oplus M_i = Y_{j-1} \oplus M_j] = 2^{-n}$ and there are $0.5\, k(k-1)/2^n$ such (i, j) pairs possible.

The probability that *bad* gets set at **line 24** of game C2 is at most $k(m-k)/2^n$. Each point X_i whose presence in the domain of η we test is of the form $X_i = Y_{i-1} \oplus M_i$, as defined at line 22. Each of these Y_{i-1} values with the exception of Y_k is uniformly selected at line 21, independent of the now-determined domain of η. As for $X_{k+1} = Y_k \oplus M_{k+1}$, the value Y_k was just selected uniformly at random (at the last execution of line 12) and is independent of the domain of η. Thus each time line 24 executes there is at most a $k/2^n$ chance that *bad* will get set, and the line is executed $m-k$ times.

The probability that *bad* gets set at **line 42** of game C2 is at most $(m'-k)/2^n$. This is because the line is executed $m'-k$ times and each time the chance that *bad* gets set is $1/2^n$ since Y'_j was just selected at random from $\{0,1\}^n$.

The probability that *bad* gets set at **line 45** of game C2 is at most $(m-k)(m'-k)$. Each point X'_j whose presence in the domain of ι is being tested is of the form $X'_j = Y'_{j-1} \oplus M'_j$, as defined at line 43. Each of these Y'_{i-1} values with the exception of Y'_k was uniformly selected at line 41, independent of the now-determined domain of ι. As for $X'_{k+1} = Y'_k \oplus M'_{k+1}$, the value Y'_k was just selected uniformly at random back at line 12 and is independent of the domain of ι apart from the domain point $Y_k \oplus M_{k+1}$, which is *guaranteed* to be distinct from $Y'_k \oplus M'_{k+1}$ by our criterion for

choosing k. Thus each time line 45 executes there is at most a $(m-k)/2^n$ chance that *bad* will get set, and the line is executed $m'-k$ times.

The probability that *bad* gets set at **line 46** of game C2 is at most $k(m'-k)$ for reasons exactly analogous to that argument used at line 24.

We conclude by the sum bound that the probability that *bad* gets set somewhere in game C2 is at most $(k(k-1)/2+k(m-k)+(m'-k)+(m-k)(m'-k)+k(m'-k))/2^n = mm'+m'-k(k-3)/2$. Since k can be zero, this value is at most $(mm'+m')/2^n$. Recalling that m' is the block length of the longer of M and M', the proof is complete. ▌

5.7 Collision-resistance under hidden-key attack

In a hidden-key attack, the adversary does not get the key K in the post-key attack phase, but instead gets an oracle for $H_K(\cdot)$. There are again three possible notions of security, analogous to those in Figure 5.3 except that, in the post-key attack phase, A is not given K but is instead given an oracle for $H_K(\cdot)$. The CR0 notion however coincides with the one for known-key attacks since by the time the post-key attack phase is reached, a cr0-adversary has already output both its points, so we get only two new notions. Formal experiments defining these two notions are given in Figure 5.7.

Definition 5.9 Let $H\colon \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Range}(H)$ be a hash function and let A be an algorithm. We let

$$\begin{aligned} \mathbf{Adv}^{\text{cr2-hk}}_H(A) &= \Pr\left[\mathbf{Exmt}^{\text{cr2-hk}}_H(A) = 1\right] \\ \mathbf{Adv}^{\text{cr1-hk}}_H(A) &= \Pr\left[\mathbf{Exmt}^{\text{cr1-hk}}_H(A) = 1\right] . \end{aligned}$$

For any t, q, μ we define

$$\begin{aligned} \mathbf{Adv}^{\text{cr2-hk}}_H(t,q,\mu) &= \max_A\{\mathbf{Adv}^{\text{cr2-hk}}_H(A)\} \\ \mathbf{Adv}^{\text{cr1-hk}}_H(t,q,\mu) &= \max_A\{\mathbf{Adv}^{\text{cr1-hk}}_H(A)\} \end{aligned}$$

where the maximum is over all A having time-complexity t, making at most q oracle queries, the sum of the lengths of these queries being at most μ. All these definitions assume adversaries are *legitimate* in the sense that the points that they query to their oracle are in the domain of the hash function. ▌

5.8 Problems

Problem 5.1 Hash functions have sometimes been constructed using a block cipher, and often this has not gone well. Let $E\colon \mathcal{K}\times\{0,1\}^n \to \{0,1\}^n$ be a block cipher

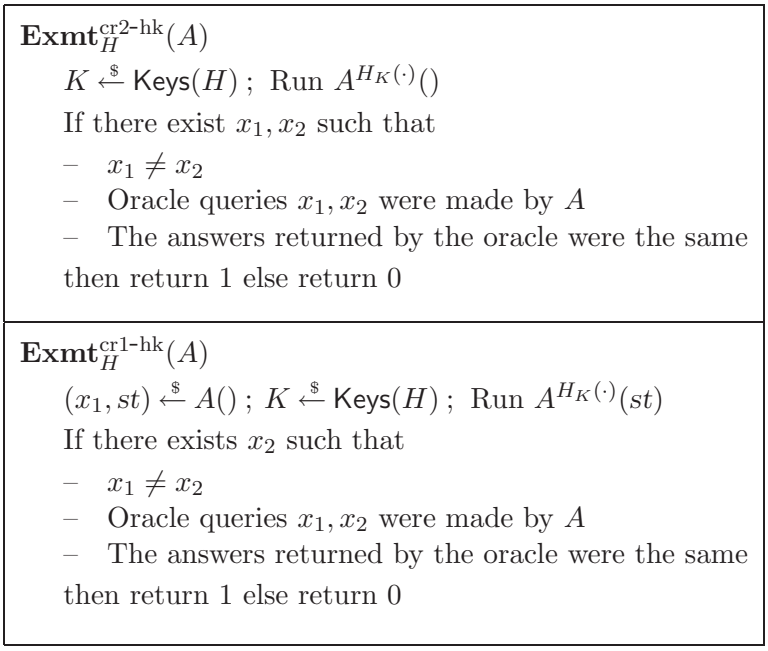

Figure 5.7: Experiments defining security notions for two kinds of collision-resistant hash functions under hidden-key attack.

and consider constructing H: $\mathcal{K} \times (\{0,1\}^n)^+ \to \{0,1\}^n$ by way of the CBC construction: let the hash of $M_1 \cdots M_m$ be Y_m where $Y_0 = 0^n$ and $Y_i = E_K(H_{i-1} \oplus M_i)$ for $i \geq 1$. Here we select K to be some public constant. Show that this hash function is not collision-resistant (no matter how good is the block cipher E).

Problem 5.2 Let $H : \mathcal{K} \times \{0,1\}^a \to \{0,1\}^n$ be an ϵ-AU hash-function family. Construct from H an ϵ-AU hash-function family $H' : \mathcal{K} \times \{0,1\}^{2a} \to \{0,1\}^{2n}$.

Problem 5.3 Let $H : \mathcal{K} \times \{0,1\}^a \to \{0,1\}^n$ be an ϵ-AU hash-function family. Construct from H an ϵ^2-AU hash-function family $H' : \mathcal{K}^2 \times \{0,1\}^a \to \{0,1\}^{2n}$.