Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?

Chun-Yuan Hsiao and Leonid Reyzin

Boston University Computer Science
111 Cummington Street
Boston MA 02215 USA
{cyhsiao,reyzin}@cs.bu.edu

June 1, 2004

Abstract

Many cryptographic primitives begin with parameter generation, which picks a primitive from a family. Such generation can use public coins (e.g., in the discrete-logarithm-based case) or secret coins (e.g., in the factoring-based case). We study the relationship between public-coin and secret-coin collision-resistant hash function families (CRHFs). Specifically, we demonstrate that:

- there is a lack of attention to the distinction between secret-coin and public-coin definitions in the literature, which has led to some problems in the case of CRHFs;
- in some cases, public-coin CRHFs can be built out of secret-coin CRHFs;
- the distinction between the two notions is meaningful, because in general secret-coin CRHFs are unlikely to imply public-coin CRHFs.

The last statement above is our main result, which states that there is no black-box reduction from public-coin CRHFs to secret-coin CRHFs. Our proof for this result, while employing oracle separations, uses a novel approach, which demonstrates that there is no black-box reduction *without* demonstrating that there is no relativizing reduction.

1 Introduction

1.1 Background

Collision-Resistant Hashing Collision-resistant (CR) hashing is one of the earliest primitives of modern cryptography, finding its first uses in digital signatures [Rab78, Rab79] and Merkle trees [Mer82, Mer89]. A hash function, of course, maps (potentially long) inputs to short outputs. Informally, a hash function is collision-resistant if it is infeasible to find two inputs that map to the same output.

It is easy to see there is no meaningful way to formalize the notion of collision-resistance for a single fixed-output-length hash function. Indeed, at least half of the 2^{161} possible 161-bit inputs to SHA-1 [NIS95] have collisions (because SHA-1 has 160-bit outputs). Hence, an algorithm finding collisions for SHA-1 is quite simple: it just has, hardwired in it, two 161-bit strings that collide. It exists, even if no one currently knows how to write it down.

Due to this simple observation, formal definitions of collision-resistant hashing (first given by Damgård [Dam87]) usually speak of collision-resistant function *families* (CRHFs).[1] A hash function family is collision-resistant if any adversary, given a function chosen randomly from the family, is unable to output a collision for it.

How to Choose from a Family? Most definitions of CRHFs do not dwell on the issue of *how* a hash function is to be chosen from a family. In this paper, we point out that this aspect of the definition is crucial. Indeed, in any application of collision-resistant hashing, some party P must choose a function from the family by flipping some random coins to produce the function description. As we demonstrate, it is important to distinguish between two cases. In the *public-coin* case these random coins can be revealed as part of the function description. In the *secret-coin* case, on the other hand, knowledge of the random coins may allow one to find collisions, and thus P must keep the coins secret after the description is produced. (For examples of both cases, see Section 2.) We note that the original definition of [Dam87] is secret-coin, and that the secret-coin definition is more general: clearly, a public-coin CRHF will also work if one chooses to keep the coins secret.

1.2 Initial Observations

Importance of the Distinction The distinction between public-coin and secret-coin CRHFs is commonly overlooked. Some works modify the secret-coin definition of [Dam87] to a public-coin definition, without explicitly mentioning the change (e.g., [BR97, Sim98]). Some definitions (e.g., [Mir01]) are ambiguous on this point. This state of affairs leads to confusion and potential problems, as discussed in three examples below.

Example 1. Some applications use the wrong definition of CRHF. For instance, in Zero-Knowledge Sets of Micali, Rabin and Kilian [MRK03], the prover uses a hash function to commit to a set. The hash function is chosen via a shared random string, which is necessary because the prover cannot be trusted to choose his own hash function (since a dishonest prover could benefit from finding collisions), and interaction with the verifier is not allowed at the commit stage (indeed, the prover does not yet know who the verifier(s) will be). In such a setting, one cannot use secret-coin CRHFs (however, in an apparent oversight, [MRK03] defines only secret-coin CRHFs). A clear distinction between public-coin and secret-coin CRHFs would make it easier to precisely state the assumptions needed in such protocols.

Example 2. The result of Simon [Sim98] seems to claim less than the proof implies. Namely, the [Sim98] theorem that one-way permutations are unlikely to imply CRHFs is stated only for public-coin CRHFs, because that is the definition [Sim98] uses. It appears to hold also for secret-coin CRHFs, but this requires re-examining the proof. Such re-examination could be avoided had the definitional confusion been resolved.

Example 3. The original result of Goldwasser and Kalai [GK03] on the security of the Fiat-Shamir transform without random oracles has a gap due to the different notions of CRHF (the gap was subsequently closed, see below). Essentially, the work first shows that if no *secret-coin* CRHFs exist, then the Fiat-Shamir transform can never work. It then proceeds to show, in a sophisticated argument, that if *public-coin* CRHFs exist, then it is possible to construct a secure identification scheme for which the Fiat-Shamir transform always results

[1] It is possible to define a single hash function (with variable output-length; cf. previous paragraph) instead of a collection of them. In this case, it can be collision-resistant only against a uniform adversary.

in an insecure signature scheme. This gap in the result would be more apparent with proper definitions.

Let us elaborate on the third example, as it was the motivating example for our work. It is not obvious how to modify the [GK03] proof to cover the case when secret-coin CRHFs exist, but public-coin ones do not. Very recently, Goldwasser and Kalai [GK] closed this gap by modifying the identification scheme of the second case to show that the Fiat-Shamir transform is insecure if secret-coin (rather than public-coin) CRHFs exist. Briefly, the modification is to let the honest prover choose the hash function during key generation (instead of the public-coin Fiat-Shamir verifier choosing it during the interaction, as in the earlier version).

Despite the quick resolution of this particular gap, it and other examples above demonstrate the importance of distinguishing between the two types of collision-resistant hashing. Of course, it is conceivable that the two types are equivalent, and the distinction between them is without a difference. We therefore set out to discover whether the distinction between public-coin and secret-coin hashing is real, i.e., whether it is possible that public-coin CRHFs do not exist, but secret-coin CRHFs do.

1.3 Our Results

Recall that public-coin hashing trivially implies secret-coin hashing. We prove the following results:

1. Dense[2] secret-coin CRHFs imply public-coin CRHFs; but
2. There is no black-box reduction from secret-coin CRHFs to public-coin CRHFs.

The first result is quite simple. The second, which is more involved, is obtained by constructing oracles that separate secret-coin CRHFs from public-coin CRHFs. Our technique for this oracle separation is different from previous separations (such as [IR89, Sim98, GKM$^+$00, GMR01, CHL02]), as explained below. We note that our second result, as most oracle separations, applies only to uniform adversaries (a notable exception to this is [GT00]).

Our results suggest that a gap between secret-coin and public-coin CRHFs exists, but only if no dense secret-coin CRHFs exist. They highlight the importance of distinguishing between the two definitions of CRHFs.

In addition to these main results, Section 5 addresses secret vs. public coins in other cryptographic primitives.

1.4 On Oracle Separations

Usually when one constructs a cryptographic primitive P (e.g., a pseudorandom generator [BM84]) out of another cryptographic primitive Q (e.g., a one-way permutation), P uses Q as a subroutine, oblivious to how Q implemented. The security proof for P usually constructs an adversary for Q using any adversary for P as a subroutine. This is known as a "*black-box reduction* from P to Q."

Note that to show that no *general* reduction from P to Q exists requires proving that Q does not exist, which is impossible given the current state of knowledge. However, it is often possible to show that no *black-box* reduction from P to Q exists; this is important because most cryptographic reductions are black-box.

[2]A CRHF is *dense* if a noticeable subset of all keys of a particular length is secure; see Section 3.

The first such statement in cryptography is due to Impagliazzo and Rudich [IR89]. Specifically, they constructed an oracle relative to which key agreement does not exist, but one-way permutations do. This means that any construction of key agreement from one-way permutations does not relativize (i.e., does not hold relative to an oracle). Hence no black-box reduction from key agreement to one-way permutations is possible, because black-box reductions relativize.

The result of [IR89] was followed by other results about "no black-box reduction from P to Q exists," for a variety of primitives P and Q (e.g., [Sim98, GKM$^+$00, GMR01, CHL02]). Most of them, except [GMR01], actually proved the slightly stronger statement that no *relativizing* reduction from P to Q exists, by using the technique of constructing an oracle.

Our proof differs from most others in that it *directly* proves that no black-box reduction exists, without proving that no relativizing reduction exists. We do so by constructing different oracles for the construction of P from Q and for the security reduction from adversary for P to adversary for Q. This proof technique seems more powerful than the one restricted to a single oracle, although it proves a slightly weaker result. The weaker result is still interesting, however, because it still rules out the most common method of cryptographic reduction. Moreover, the stronger proof technique may yield separations that have not been achievable before.

We note that [GMR01] also directly prove that no black-box reduction exists, without proving that no relativizing reduction exists. Our approach is different from [GMR01], whose approach is to show that for every reduction, there is an oracle relative to which this reduction fails.

For a detailed discussion on black-box reductions, see [RTV04]. All reductions in this paper are what they refer to as *fully black-box* reductions.

2 Definitions of Public-Coin and Secret-Coin CRHFs

Examples Before we define public-coin and secret-coin hashing formally, consider the following two example hash function families. The first one, keyed by a prime p with a large prime $q|(p-1)$, and two elements $g, h \in \mathbb{Z}_p^*$ of order q, computes $H_{p,g,h}(m) = g^{m_1}h^{m_2}$, where m_1 and m_2 are two halves of m (here we think of m as an element of $Z_q \times Z_q$).[3] The second one, keyed by a product n of two primes $p_1 \equiv 3 \pmod 8$, and $p_2 \equiv 7 \pmod 8$ and a value $r \in \mathbb{Z}_n^*$, computes $H_{n,r}(m) = 4^m r^{2^{|m|}} \bmod n$.[4]

The first hash function family is secure as long as discrete logarithm is hard. Thus, if one publishes the random coins used to generate p, g and h, the hash function remain secure (as long as the generation algorithm doesn't do anything esoteric, such as computing h as a random power of g). On the other hand, the second hash function family is secure based on factoring, and is entirely insecure if the factors of n are known. Thus, publishing the random coins used to generate p_1 and p_2 renders the hash function insecure, and the coins must be kept secret.[5]

Definitions We say that a function is *negligible* if it vanishes faster than any inverse polynomial. We let PPTM stand for a probabilistic polynomial-time Turing machine. We use $M^?$ to denote an oracle Turing machine, and M^A to denote M instantiated with oracle A.

Let k be the security parameter, and let ℓ be a (length) function that does not expand or shrink its input more than a polynomial amount. Below we define two kinds of CRHFs: namely, secret-

[3]This family is derived from Pedersen commitments [Ped91].

[4]This is essentially the construction of [Dam87] based on the claw-free permutations of [GMR88].

[5]It should be noted, of course, whether it is secure to publish the coins depends not only on the family, but also on the key generating algorithm itself: indeed, the first family can be made insecure if the coins are used to generate h as a power of g, rather than pick h directly. Likewise, the second family could be made secure if it were possible to generate n "directly," without revealing p_1 and p_2 (we are not aware of an algorithm to do so, however).

coin and public-coin. The secret-coin CRHFs definition is originally due to Damgård [Dam87], and the definition here is adapted from [Rus95].

Definition 1. A *Secret-Coin* Collision Resistant Hash Family is a collection of functions $\{h_i\}_{i \in I}$ for some index set $I \subseteq \{0,1\}^*$, where $h_i : \{0,1\}^{|i|+1} \to \{0,1\}^{|i|}$, and

1. There exists a PPTM GEN, called the generating algorithm, so that $\mathsf{GEN}(1^k) \in \{0,1\}^{\ell(k)} \cap I$.
2. There exists a PPTM EVA, called the function evaluation algorithm, so that $\forall i \in I$ and $\forall x \in \{0,1\}^{|i|+1}$, $\mathsf{EVA}(i,x) = h_i(x)$.
3. For all PPTM ADV, the probability that $\mathsf{ADV}(i)$ outputs a pair (x,y) such that $h_i(x) = h_i(y)$ is negligible in k, where the probability is taken over the random choices of GEN in generating i and the random choices of ADV.

Definition 2. A *Public-Coin* Collision Resistant Hash Family is a collection of functions $\{h_i\}_{i \in \{0,1\}^*}$, where $h_i : \{0,1\}^{\ell(|i|)+1} \to \{0,1\}^{\ell(|i|)}$, and

1. A PPTM GEN on input 1^k outputs a uniformly distributed string i of length k.
2. There exists a PPTM EVA, called the function evaluation algorithm, so that $\forall i \in \{0,1\}^*$ and $\forall x \in \{0,1\}^{\ell(|i|)+1}$, $\mathsf{EVA}(i,x) = h_i(x)$.
3. For all PPTM ADV, the probability that $\mathsf{ADV}(i)$ outputs a pair (x,y) such that $h_i(x) = h_i(y)$ is negligible in k, where the probability is taken over the random choices of GEN in generating i and the random choices of ADV.

A pair (x,y) such that $h_i(x) = h_i(y)$ is called a *collision* for h_i.

Remarks The generating algorithm in the public-coin case is trivially satisfied. We keep it here for comparison with the secret-coin case. Note that in both cases, on security parameter k, GEN outputs a function that maps $\{0,1\}^{\ell(k)+1}$ to $\{0,1\}^{\ell(k)}$. This may seem restrictive as the hash functions only compress one bit. However, it is easy to see that h_i can be extended to $\{0,1\}^n$ for any n, and remain collision-resistant with $\ell(k)$-bit outputs, by the following construction: $h_i^*(x) = h_i(\ldots h_i(h_i(h_i(x_1 \circ x_2 \circ \ldots \circ x_{\ell(k)+1}) \circ x_{\ell(k)+2}) \circ x_{\ell(k)+3}) \ldots \circ x_n)$, where x_j denotes the j-th bit of the input string x.

3 Dense Secret-Coin CRHFs imply Public-Coin CRHFs

The notion of *dense* public-key cryptosystems was introduced by De Santis and Persiano in [DP92]. By "dense" they mean that a uniformly distributed string, with some noticeable probability, is a secure public key. We adapt the notion of denseness in public-key cryptosystems from [DP92] to the context of CRHFs. Informally, a d-dense secret-coin CRHF is a secret-coin CRHF with the following additional property: if we pick a k-bit string at random, then we have probability at least k^{-d} of picking an index i for a *collision-resistant* function.[6]

Note that, for example, the factoring-based secret-coin CRHF from Section 2 is dense, because the proportion of k-bit integers that are products of two equal-length primes is $\Theta(k^{-2})$. In fact, we are not aware of any natural examples of secret-coin CRHFs that are not dense (artificial examples, however, are easy to construct).

[6]Confusingly, sometimes the term dense is used to denote a function family where each function has a dense domain, e.g., [Hai04]. This is unrelated to our use of the term.

Given a d-dense secret-coin CRHF, if we pick k^{d+1} strings of length k at random, then with high probability, at least one of them defines a collision-resistant hash function.

Hence, we can build a public-coin CRHF from such dense secret-coin CRHF as follows.

1. Generate k^{d+1} random k-bit strings, independently. These strings specify k^{d+1} hash functions $h_1, h_2, \dots h_{k^{d+1}}$ in the secret-coin CRHF (strictly speaking, some strings may not define functions at all, because they are not produced by GEN; however, simply define $h_i(x) = 0^{\ell(k)}$ if $\mathsf{EVA}(i, x)$ does not produce an output of length k in the requisite number of steps).

2. Through the construction described in Section 2, extend the domain of each of these function to binary strings of length $\ell(k)k^{d+1} + 1$. Let the resulting functions be $h_1^*, \dots, h_{k^{d+1}}^*$.

3. On an input x of length $\ell(k)k^{d+1} + 1$, output concatenation of $h_1^*(x), h_2^*(x), \dots, h_{k^{d+1}}^*(x)$.

The resulting hash maps binary strings of length $\ell(k)k^{d+1} + 1$ to binary strings of length $\ell(k)k^{d+1}$, and is collision-resistant because at least one of $h_1^*, h_2^*, \dots, h_{k^{d+1}}^*$ is. (If an adversary could find a collision in the resulting hash function, then the same collision would work for collision-resistant hash function among $h_1^*, h_2^*, \dots, h_{k^{d+1}}^*$, immediately leading to a contradiction.)

The above discussion yields the following theorem.

Theorem 1. *The existence of dense secret-coin CRHF implies the existence of public-coin CRHF.*

4 Separating Public-Coin CRHFs from Secret-Coin CRHFs

4.1 Black-Box Reductions

Impagliazzo and Rudich [IR89] provided an informal definition of black-box reductions, and Gertner et al. [GKM$^+$00] formalized it. We recall their formalization.

Definition 3. A *black-box reduction* from primitive P to primitive Q consists of two oracle PPTMs M and A_Q satisfying the following two conditions:

If Q can be implemented, so can P: $\forall N$ (not necessarily PPTM) implementing Q, M^N implements P; and

If P is broken, so is Q: $\forall A_P$ (not necessarily PPTM) breaking M^N (as an implementation of P), $A_Q^{A_P, N}$ breaks N (as an implementation of Q).

The first condition is only a functional requirement; i.e., the term "implement" says nothing about security, but merely says an algorithm satisfies the syntax of the primitive.

4.2 The Main Result

Theorem 2. *There is no black-box reduction from public-coin CRHF to secret-coin CRHF.*

Proof. The following proposition is at the heart of our approach: it shows that it is sufficient to construct different oracles F and G, such that G is used in the *implementations*, while F and G are used for the *adversaries*. This is in contrast to the single-oracle approach usually taken to prove black-box separations.

Proposition 1. *To show that there is no black-box reduction from* public-coin *collision resistant hashing (P) to* secret-coin *collision resistant hashing (Q), it suffices to construct two oracles* F *and* G *such that,*

1. *there is an oracle PPTM L such that $N = L^{\mathsf{G}}$ implements secret-coin hashing;*
2. *for all oracle PPTM M, if M^{G} implements public-coin hashing, then there exists a probabilistic polynomial time adversary A such that $A_P = A^{\mathsf{F}}$ finds a collision for M^{G};*
3. *there is no oracle PPTM B such that $B^{\mathsf{F},\mathsf{G}}$ finds a collision for N.*

Proof. To show that there is no black-box reduction from *public-coin* collision resistant hashing (P) to *secret-coin* collision resistant hashing (Q), we need to negate the definition of black-box reduction from Section 2; i.e., we need to show that for every oracle PPTMs M and A_Q,

Q **can be implemented:** $\exists N$ that implements Q, and if M^N implements P, then

P **can be broken, without breaking** Q**:** $\exists A_P$ that breaks M^N (as an implementation of P), while $A_Q^{A_P,N}$ does not break N (as an implementation of Q).

Recall that "implement" here has only functional meaning.

The first condition clearly implies that Q can be implemented. The second condition also clearly implies that P can be broken: one simply observes that $M^N = M^{L^{\mathsf{G}}}$, and L is a PPTM; hence, writing M^{G} is equivalent to writing M^N. The third condition implies that P can be broken without breaking Q, essentially because Q can never be broken. More precisely, the third condition is actually stronger than what we need: all we need is that for each A_Q, there is A_P that breaks M^N, while $A_Q^{A_P,N}$ does not break N. Instead, we will show that a single A_P essentially works for all A_Q: namely, $A_P = A^{\mathsf{F}}$, for a fixed oracle F and a polynomial-time A. Such A_P breaks M^N; however, as condition 3 in the proposition statement implies, $A_Q^{A_P,N}$ will be unable to break N, because $A_Q^{A_P,N} = A_Q^{A^{\mathsf{F}},L^{\mathsf{G}}} = B^{\mathsf{F},\mathsf{G}}$ for some oracle PPTM B. $\square$

Remarks Note that if the implementation has access to not only G but also F, it becomes the usual single-oracle separation. The reason why we do not give the implementation access to F is to avoid "self-referencing" when defining F. To see this, note that F is the "collision finder" and is defined according to the oracles that the implementation has access to.[7]

The rest of this section is devoted to constructing such F and G and proving that they work.

4.3 The Oracles F and G

In constructing F and G, we will use the Borel-Cantelli Lemma (see, e.g., [AG96]), which states that if the sum of the probabilities of a sequence of events converges, then the probability that infinitely many of these events happen is zero. Formally,

Lemma 3 (Borel-Cantelli Lemma). *Let $B_1, B_2, \ldots$ be a sequence of events on the same probability space. Then $\sum_{n=1}^{\infty} \mathbf{Pr}[B_n] < \infty$ implies $\mathbf{Pr}[\bigwedge_{k=1}^{\infty} \bigvee_{n\geq k} B_n] = 0$.*

[7]Similar concern occurs in [Sim98], where constructing the collision-finder requires more careful design.

We first construct "random" F (collision-finder) and G (secret-coin hash), and then use the above lemma to show that at least one pair of F and G works.

Intuitively, we want F to break any public-coin hashing but not break some secret-coin hashing. More precisely, F will find a collision if it is supplied with the coins of the generating algorithm and will refuse to do so without the coins.

- G consists of two collections of functions $\{g_i\}_{i \in \mathbb{N}}$ and $\{h_\alpha\}_{\alpha \in \{0,1\}^*}$, where each g_i is a random function from $\{0,1\}^i$ to $\{0,1\}^{2i}$. We will call a binary string *valid* if it is in the range of g, and *invalid* if not. Each h_α is a random function from $\{0,1\}^{|\alpha|+1}$ to $\{0,1\}^{|\alpha|}$ if α is valid, and is a constant function $0^{|\alpha|}$ if α is invalid. We will call queries to h_α valid (resp. invalid) if α is valid (resp. invalid).

- F takes a deterministic oracle machine $M^?$ and 1^ℓ as input, and outputs a collision of length $\ell+1$ for M^{G} if M^{G} satisfies the following conditions.

  1. M^{G} maps $\{0,1\}^{\ell+1}$ to $\{0,1\}^\ell$.
  2. M^{G} never queries h_α for some α not obtained by previously querying g. I.e., whenever M^{G} queries h_α, this α is the answer to some g-query that M^{G} has previously asked.

  When both conditions hold, F picks a random x from $\{0,1\}^{\ell+1}$ that has a collision, then a random y ($\neq x$) that collides to x (i.e., $M^{\mathsf{G}}(x) = M^{\mathsf{G}}(y)$), and outputs (x, y). Otherwise F outputs $\perp$.

  Observe that when F outputs (x, y), not only x, but also y is uniformly distributed over all points that have a collision. Indeed, let C be the total number of points that have a collision, and suppose y has c collisions $(x_1, x_2, \ldots, x_c)$: then $\mathbf{Pr}[y \text{ is chosen}] = \sum_{i=1}^{c} 1/c\, \mathbf{Pr}[x_i \text{ is chosen}] = 1/c \cdot (c/C) = 1/C$.

Remarks The reason for g being length-doubling is to have a "sparse" function family. More specifically, it should be hard to get a value in the range of g without applying it.

As in [Sim98], there are various ways of constructing F (the collision-finding oracle): one can choose a random pair that collides, or a random x then a random y (possibly equal to x) that collides to x. The second construction has the advantage, in analysis, that both x and y are uniformly distributed but does not always give a "correct" collision, like the first one does. Our F has both properties.

4.4 Secret-Coin Collision-Resistant Hash Family Based on G

In this section we construct a secret-coin CRHF. The construction is straightforward given the oracle G: the generating algorithm uses g and the hashing uses h. More precisely, on input 1^k the generating algorithm picks a random seed $r \in \{0,1\}^k$ and outputs $\alpha = g_k(r)$. The hash function is h_α. Note that the adversary A (who is trying to find a collision) is given only α but not r. We will show that for measure one of oracles F and G, the probability over r and A's coin tosses that A finds a collision for h_α is negligible. Recall that A has access to both F and G.

Define D as the event that A outputs a collision for h_α in the following experiment:

$$r \leftarrow_R \{0,1\}^k,\ \alpha \leftarrow g_k(r),\ (x, y) \leftarrow A^{\mathsf{F},\mathsf{G}}(\alpha).$$

And in the same experiment, define B as the event that during its computation, A queries F on $M^?$, where $M^?$ is some deterministic oracle machine that queries its oracle on a preimage of α

under g_k (i.e., intuitively, $M^?$ has r hardwired in it). Suppose A's running time is bounded by k^c for some constant c. The probability that B happens is at most the probability of inverting the random function g_k. If α has a unique preimage, this is at most $k^c/2^k$; the probability that α has two or more preimages is at most $1/2^k$ (because it's the probability that r collides with another value under g_k); hence $\mathbf{Pr}[B] \leq (k^c+1)/2^k$. The probability that D happens conditioned on $\neg B$ is at most the probability of finding a collision for random function h_α, which is bounded by $k^{2c}/2^{2k}$. Recall that A can be randomized. We thus have

$$\begin{aligned}
\underset{\mathsf{F},\mathsf{G},r,A}{\mathbf{Pr}}[D] &= \mathbf{Pr}[B]\cdot\mathbf{Pr}[D|B] + \mathbf{Pr}[\neg B]\cdot\mathbf{Pr}[D|\neg B] \\
&\leq \mathbf{Pr}[B] + \mathbf{Pr}[D|\neg B] \\
&\leq (k^c+1)/2^k + k^{2c}/2^{2k} \\
&\leq 2k^c/2^k\,.
\end{aligned}$$

By the Markov inequality, $\mathbf{Pr}_{\mathsf{F},\mathsf{G}}[\mathbf{Pr}_{r,A}[D] \geq k^2 \cdot 2k^c/2^k] \leq 1/k^2$. Since $\sum_k 1/k^2$ converges, the Borel-Cantelli lemma implies that for only measure zero of F and G, can there be infinitely many k for which event D happens with probability (over r and A's coins) greater than or equal to $k^{c+2}/2^{k-1}$. This implies that for measure one of F and G, event D happens with probability (over r and A's coins) smaller than $k^{c+2}/2^{k-1}$ (a negligible function) for all large enough k. There are only countably many adversaries A, so we have the following lemma.

Lemma 4. *For measure one of* F *and* G*, there is a CRHF using* G*, which is secure against adversaries using* G *and* F.

4.5 No Public-Coin Collision-Resistant Hash Family Based on G

In this section we show that any implementation of public-coin hashing using oracle G cannot be collision-resistant against adversaries with oracle access to both F and G.[8] More precisely, let $r \in \{0,1\}^k$ be the public randomness used by the generating algorithm for a family of hash functions, and let $M^?$ be the evaluation algorithm. I.e., $M^{\mathsf{G}}(r,\cdot)$ is the hash function specified by r. Assume that $M_r^{\mathsf{G}}(\cdot) \triangleq M^{\mathsf{G}}(r,\cdot)$ maps $\{0,1\}^{\ell(k)+1}$ to $\{0,1\}^{\ell(k)}$, where ℓ is a function that does not expand or shrink the input by more than a polynomial amount. We will show how to find x and y of length $\ell(k)+1$ such that $M_r^{\mathsf{G}}(x) = M_r^{\mathsf{G}}(y)$.

An immediate attempt is to query $\mathsf{F}(M_r^?, 1^{\ell(k)})$, but notice that M_r^{G} may query h_α for arbitrary α,[9] which prevents F from finding a collision for us. However, these α are likely to be invalid, and hence oracle answers to these queries are likely to be $0^{|\alpha|}$. So we can construct a machine $\tilde{M}_r^?$ that behaves "similar" to $M_r^?$ but only after getting α from g does it query h_α. And instead of finding collision for M_r^{G}, we find collision for $\tilde{M}_r^{\mathsf{G}}$, which can be done by simply querying $F(\tilde{M}_r^?, 1^{\ell(k)})$.

Suppose the running time of M_r^{G} is bounded by k^c for some constant $c > 1$. Before simulating M_r^{G}, $\tilde{M}_r^{\mathsf{G}}$ queries g on all inputs of length smaller than or equal to $4c\log k$. This takes $2k^{4c}$ steps. Now $\tilde{M}_r^{\mathsf{G}}$ simulates M_r^{G} step by step, except for queries to h_α. If α is the answer to one of the queries $\tilde{M}_r^{\mathsf{G}}$ already asked of G (either before the beginning of the simulation or when simulating M_r^{G}), then $\tilde{M}_r^{\mathsf{G}}$ actually queries h_α. Else it returns $0^{|\alpha|}$ as the answer to M_r^{G} without querying h_α.

Now fix r and x. For every $M^?$ the probability, over random G, that $\tilde{M}_r^{\mathsf{G}}(x) \neq M_r^{\mathsf{G}}(x)$ is at most the probability, over G, that M_r^{G} queries h_α for some valid α of length greater than $8c\log k$

[8] In fact, only F is needed to find a collision.

[9] In particular, those α not obtained by previously querying g.

without receiving it from g.[10] Consider the very first time that M_r^{G} makes such a "long" valid query. Let n_g be the number of queries to g on inputs longer than $4c\log k$, and n_h be the number of invalid queries to h prior to this point. Then the probability in question is upper bounded by $k^c \cdot \frac{k^{4c}-n_g-n_h}{k^{8c}-n_g}$, which is at most $1/k^{3c}$. For every fixed G and r, call an x "bad" if $\tilde{M}_r^{\mathsf{G}}(x) \neq M_r^{\mathsf{G}}(x)$. We have

$$\mathop{\mathbf{Ex}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{x}[x \text{ is bad}]] = \mathop{\mathbf{Pr}}_{\mathsf{G},x}[x \text{ is bad}] \leq 1/k^{3c}.$$

Next, notice that there are at most half of x that have no collisions, and F would pick its answer $(x_{\mathsf{F}}, y_{\mathsf{F}})$, uniformly, from those points that have a collision. So for a fixed G, the probability over F that x_{F} is bad is at most twice the probability over random $x \in \{0,1\}^{\ell(k)+1}$ that x is bad. Also recall that the distribution of y_{F} is the same as x_{F}. So for every $M^?$,

$$\mathop{\mathbf{Ex}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{\mathsf{F}}[\text{at least one of } (x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is bad}]] \leq 4 \cdot \mathop{\mathbf{Ex}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{x}[x \text{ is bad}]].$$

If none of $(x_{\mathsf{F}}, y_{\mathsf{F}})$ is bad, this pair would be a collision not only for $\tilde{M}_r^{\mathsf{G}}$ but also for M_r^{G}. We have

$$\mathop{\mathbf{Pr}}_{F,\mathsf{G},r}[(x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is not a collision of } M_r^{\mathsf{G}}] \leq 4 \mathop{\mathbf{Pr}}_{\mathsf{G},x,r}[x \text{ is bad}] \leq 4/k^{3c},$$

then

$$\mathop{\mathbf{Pr}}_{\mathsf{F},\mathsf{G}}[\mathop{\mathbf{Pr}}_{r}[(x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is not a collision of } M_r^{\mathsf{G}}] \geq 4/k^c] \leq 1/k^{2c}.$$

Since $\sum_k 1/k^{2c}$ converges, the Borel-Cantelli lemma implies that for only measure zero of F and G, can we have $\mathbf{Pr}_r[(x_{\mathsf{F}}, y_{\mathsf{F}})$ is not a collision of $M_r^{\mathsf{G}}] \geq 4/k^c$ for infinitely many k. In other words, for measure one of F and G, $\mathbf{Pr}_r[(x_{\mathsf{F}}, y_{\mathsf{F}})$ is a collision of $M_r^{\mathsf{G}}] \geq 4/k^c$ for all large enough k. There are only countably many oracle machines $M^?$, each of which can be collision resistant for only measure zero of F and G. We conclude the following.

Lemma 5. *For measure one of* F *and* G*, any implementation of public-coin hash function families using* G *cannot be collision-resistant against adversaries using* F.

This conclude the proof of Theorem 2. □

5 Public Coins vs. Secret Coins For Other Primitives

Perhaps the lack of attention in the literature to the distinction between secret- and public-coin primitives is due, in part, to the fact that this distinction is often not meaningful.

For example, for one-way function families, these two notions are equivalent, because a secret-coin one-way function family implies a single one-way function (which trivially implies a public-coin one-way function family). Indeed, take the generating algorithm g and evaluation algorithm f and define $F(r, x) \triangleq (g(r), f_{g(r)}(x))$; this is one-way because an adversary who can come up with (r', x') such that $g(r) = g(r')$ and $f_{g(r')}(x') = f_{g(r)}(x)$ can be directly used to invert $f_{g(r)}(x)$, since $f_{g(r)}(x') = f_{g(r')}(x') = f_{g(r)}(x)$.

On the other hand, for trapdoor permutations (and public-key schemes), the notion of public-coin generation is meaningless: indeed the trapdoor (or the secret key) must be kept secret.

However, it seems that this distinction is interesting for some primitives in addition to collision-resistant hash functions. The relationships between public-coin and secret-coin versions of one-way

[10]Recall that g is length-doubling.

permutation families and claw-free permutation families are unknown.[11] In particular, claw-free permutations are related to collision-resistant hashing [Dam87, Rus95], which suggests that the distinction for claw-free permutations is related to the distinction for CRHFs.

Acknowledgments. We thank Yael Tauman Kalai for many helpful discussions, and Ron Rivest for assistance with the history of hashing. Thanks also to the anonymous referees for insightful comments. This work was funded in part by the National Science Foundation under Grant No. CCR-0311485.

References

[AG96] Malcolm Adams and Victor Guillemin. *Measure Theory and Probability.* Springer Verlag, 1996.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, November 1984.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 17–21 August 1997.

[CHL02] Yan-Cheng Chang, Chun-Yun Hsiao, and Chi-Jen Lu. On the imposibilities of basing one-way permutations on central cryptographic primitives. In Yuliang Zheng, editor, *Advances in Cryptology—ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 110–124, Queenstown, New Zealand, 1–5 December 2002. Springer-Verlag.

[Dam87] Ivan Damgård. Collision-free hash functions and public-key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988, 13–15 April 1987.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[GK] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. Available From `http://www.mit.edu/~tauman/`.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science* [IEE03].

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science* [IEE00], pages 325–335.

[11] We believe that the same construction of F and G (up to slight modifications) separates public-coin and secret-coin one-way permutation families.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, October 2001. IEEE.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science* [IEE00].

[Hai04] Iftach Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In Naor [Nao04], pages 394–409.

[IEE00] IEEE. *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, California, November 2000.

[IEE03] IEEE. *44th Annual Symposium on Foundations of Computer Science*, Cambridge, Massachusetts, October 2003.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, Washington, 15–17 May 1989.

[Mer82] Ralph C. Merkle. *Secrecy, Authentication, and Public Key Systems.* UMI Research Press, 1982.

[Mer89] Ralph C. Merkle. A certified digital signature. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990, 20–24 August 1989.

[Mir01] Ilya Mironov. Hash functions: From merkle-damgrd to shoup. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 166–181. Springer-Verlag, 19–23 August 2001.

[MRK03] Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Annual Symposium on Foundations of Computer Science* [IEE03], pages 80–91.

[Nao04] Moni Naor, editor. *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*. Springer-Verlag, February 19–21 2004.

[NIS95] FIPS publication 180-1: Secure hash standard, April 1995. Available from `http://csrc.nist.gov/fips/`.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992, 11–15 August 1991.

[Rab78] Michael O. Rabin. Digitalized signatures. In Richard A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[Rab79] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge, MA, January 1979.

[RTV04] Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of reducibility between cryptographic primitives. In Naor [Nao04], pages 1–20.

[Rus95] A. Russell. Necessary and sufficient conditions for collision-free hashing. *Journal of Cryptology*, 8(2):87–100, 1995.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*. Springer-Verlag, May 31–June 4 1998.