The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function

John Black

Dept. of Computer Science, University of Colorado, Boulder CO 80309, USA,
jrblack@cs.colorado.edu, www.cs.colorado.edu/~jrblack

Abstract. The Ideal-Cipher Model of a blockcipher is a well-known and widely-used model dating back to Shannon [25] and has seen frequent use in proving the security of various cryptographic objects and protocols. But very little discussion has transpired regarding the meaning of proofs conducted in this model or regarding the model's validity. In this paper, we briefly discuss the implications of proofs done in the ideal-cipher model, then show some limitations of the model analogous to recent work regarding the Random-Oracle Model [2]. In particular, we extend work by Canetti, Goldreich and Halevi [5], and a recent simplification by Maurer, Renner, and Holenstein [15], to exhibit a blockcipher-based hash function that is provably-secure in the ideal-cipher model but trivially insecure when instantiated by any blockcipher.

Keywords: Ideal-Cipher Model, Information-Theoretic Cryptography, Random-Oracle Model, Uninstantiability

1 Introduction

The Standard Model. Before we can prove the security of a cryptographic system or object, we must specify what model we are using. The most common model used in modern cryptography is the so-called "standard model." Here we use no special mathematical objects such as infinite random strings or random oracles [2], and we abstract our communications system typically as a reliable but insecure channel. We have not been able to achieve most common cryptographic goals in the standard model without making additional complexity-theoretic hardness assumptions, because we still have no proof that any of our standard cryptographic building blocks have computational lower bounds. The common assumptions are typically that factoring the product of large primes is hard, or that discrete log is intractible in certain sufficiently large groups, or that AES is a good pseudo-random permutation (PRP) [16]. The standard model is usually well-accepted in our community despite the fact that proofs done in this model rest upon unproven assumptions and that already much relevant real-world context has been abstracted away (timing, power consumption, error messages, and other real-world effects are typically not included as part of the model in spite of the demonstrated fact they are often relevant to security).

The Random-Oracle Model. When proofs in the standard model are unappealing or are provably impossible (eg, see [19]), we often resort to proofs using an alternative model. By far the best-known is the "Random-Oracle Model." The random-oracle model was used for some time before being formalized by Bellare and Rogaway [2], and continues to see widespread use today (there are more than a hundred instances; for a few examples see [2, 11, 18, 21, 24]). In the random-oracle model we have a public random function, accessible to all parties, which typically accepts any string from $\{0,1\}^*$ and outputs n bits. For each element in its domain, the corresponding n-bit output is uniform and independent from all other outputs. Proofs conducted in the random-oracle model often admit schemes which are provably-secure and more efficient than schemes which have been proven secure in the standard model, and for this reason the random-oracle model has been widely-adopted.

Of course random oracles do not exist in practice, and if the schemes proven secure in the random-oracle model are going to be put into use, we must choose some object to implement the random oracle. This step is called "instantiation." Most often, random oracles are instantiated with cryptographic hash functions such as SHA-1 [20]. The following question then arises: now that we have instantiated our random oracle with a concrete function, what security guarantees do we have? Does our proof in the random-oracle model have any bearing on the security of the instantiated system?

For quite some time there has been concern in our community that the random-oracle model should be treated with suspicion, and proofs in the standard model should be preferred. As a recent example, the main selling point of the Cramer-Shoup cryptosystem [7] is that it is provably-secure in the standard model and still practical (and, as with most proofs in the standard model, comes with an assumption: the Decisional Diffie-Hellman assumption [4]). Further doubt has been recently cast on the random-oracle model due to a string of results exhibiting schemes which are provably-secure in the random-oracle model but are completely insecure when instantiated by *any* hash function [1, 5, 6, 15]. Schemes of this type are called "uninstantiable."

It has been noted [2] that proofs done in the random-oracle model *do* guarantee one thing: if the adversary treats the instantiated random oracle as a black box, promising not to think about its inner workings, promising not to exploit any unnatural behavior related to the fact that we have instantiated with some algorithm that has a compact description, then the proof remains valid in the standard model. Of course there is no guarantee that real adversaries would abide by such restrictions, and indeed they would be remiss if they did. Nonetheless, no scheme has thus far been proven secure in the random-oracle model and then broken once instantiated, unless this was the goal from the start.

The Ideal-Cipher Model. Blockciphers are a common building block for cryptographic protocols. In the standard model the associated assumption for blockciphers is that they are "pseudo-random permutations" (PRPs). By this we mean (informally) that an n-bit blockcipher under a secret randomly-chosen key is computationally indistinguishable from a randomly-chosen n-bit permutation. Proofs conducted using this assumption typically give reductions showing that if an adversary breaks some scheme, then there exists an associated adversary that can efficiently distinguish the underlying blockcipher from random.

There are countless examples where the PRP assumption in the standard model is sufficient, but there are also plenty of cases where we cannot get a proof to go through. In certain cases it can be shown that blockcipher-based schemes we believe to be secure cannot have a proof of security using only the PRP assumption in the standard model [26]. In this case we are faced with either abandoning attempts at a proof, or using an alternate model.

The blockcipher analog for the random-oracle model is variously called the "Shannon Model," the "Black-Box Model," or the "Ideal-Cipher Model." We will prefer the latter name in this paper.

Though not as widely-used as the random-oracle model, the ideal-cipher model dates back to Shannon [25] and has been used in a variety of settings (see, for example [3, 9, 10, 12, 13, 17, 27]). In the ideal-cipher model we think of a blockcipher E with k-bit key and n-bit blocksize as being chosen uniformly from the set of all possible blockciphers of this form. For each key, there are $2^n!$ permutations, and since any permutation may be assigned to a given key, there are $(2^n!)^{2^k}$ possible blockciphers. When we instantiate our black box, it becomes some particular blockcipher. AES with a 128-bit key is one choice from the nearly $2^{2^{263}}$ blockciphers we could have chosen (though in the spirit of Kolmogorov complexity and in line with the main result of this paper, we should note that the vast majority of these blockciphers will not have an efficient and compact C implementation).

The ideal-cipher model is analogous to the random-oracle model with three notable exceptions:

- The ideal cipher has a permutivity requirement that random oracles obviously do not.
- Adversaries interacting with an ideal-cipher oracle are typically given access to both the cipher and its inverse.
- The blocksize n of the ideal cipher is typically fixed a priori. This means that an ideal cipher is a finite object while the random oracle is an infinite one.

The ideal-cipher model has been used in a variety of settings, and like the random-oracle model, some researchers question the wisdom of its use. The argument is completely analogous: if a scheme is proved secure in the ideal-cipher model, what exactly are we guaranteed once the ideal cipher is instantiated by a real blockcipher? And if the answer is essentially "not much," then what is the value of such proofs? A common argument against the ideal-cipher model is that most real-world blockciphers have distinguishing patterns which would exist with exceedingly small probability in a collection of random permutations. The key complementation property of DES is a typical example of this [16]. Although no such properties are currently known for AES, some blockcipher experts who are comfortable with the assumption that AES is a good PRP are reluctant to model AES as ideal because of practical concerns: the AES key schedule, for instance, is quite simple and it perhaps contains related-key properties we have not yet discovered.

As compensation to the adversary for his respecting the blockcipher as a black box, we often endow him with limitless computational resources. In this respect, many proofs done in the ideal-cipher model are information theoretic. This too is unrealistic, but here we are giving the *adversary* more power rather than enhancing the *objects* themselves. Nonetheless, it is saying something about the strength of our model that it allows us to achieve information-theoretic security.

The main result of this paper is to exhibit a blockcipher-based hash function that is secure in the ideal-cipher model against information-theoretic adversaries but which is trivially insecure once instantiated with any blockcipher. In order to state this result more clearly, we take a short detour to review blockcipher-based hash functions.

Blockcipher-Based Hash Functions. One area of recently-renewed interest involves constructing hash functions from blockciphers. This approach, dating back at least to Rabin [23], uses some blockcipher E with an n-bit key and an n-bit blocksize, and builds a compression function from it. Iterating this function then hopefully produces a collision-resistant hash function. Preneel, Govaerts, and Vandewalle [22] conducted a systematic study of a class of 64 blockcipher-based hash functions. They focused on compression functions of the form $f(h_{i-1}, m_i) = E_a(b) \oplus c$ where $a, b, c \in \{h_{i-1},\ m_i,\ h_{i-1} \oplus m_i,\ v\}$ for some fixed constant v. We can now hash any $M \in (\{0,1\}^n)^+$ by writing $M = M_1 \cdots M_\ell$ and then setting h_0 to some constant (typically 0^n) and setting $h_i = f(h_{i-1}, m)$. We return h_ℓ as the digest. The PGV analysis consisted of testing a series of attacks on each of these iterated hash functions. Black, Rogaway and Shrimpton [3] considered these same 64 constructions exhibiting either an attack or a proof of security (in the ideal-cipher model) for each. They determined that 20 of the 64 schemes were provably collision-resistant up to the birthday bound. For one example, see Figure 1.

Although a proof of security for a blockcipher-based hash function in the *standard model* would be prefered, it has been shown that the PRP assumption is insufficient for building a collision-resistant hash function [26]. Indeed, one can easily imagine a blockcipher $\widetilde{E} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ that is a good PRP, but which fails when used in the MMO construction of Figure 1. For example, let blockcipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a good PRP and consider blockcipher $\widetilde{E}$ defined as follows:

$$\widetilde{E}(K,X) = \begin{cases} K & \text{if } X = K \\ E(K,K) & \text{if } X = E^{-1}(K,K) \\ E(K,X) & \text{otherwise} \end{cases}$$

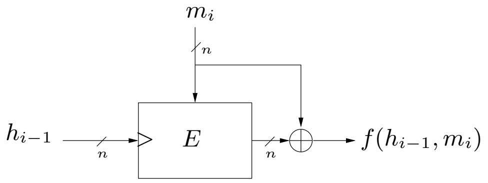

Fig. 1. The Matyas-Meyer-Oseas (MMO) compression function [14], called H_1 in [3]. $E\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher; the hatch mark denotes the location of the key. Iterating this compression function results in a provably-secure blockcipher-based hash function in the ideal-cipher model.

So $\widetilde{E}$ is the same blockcipher as E with one change: we now have the invariant that $E(K, K) = K$ for all $K \in \{0,1\}^n$. Clearly $\widetilde{E}$ is a good PRP since E was: for a *randomly*-chosen key K, $\widetilde{E}(K, \cdot)$ is computationally indistinguishable from a random permutation. However, using $\widetilde{E}$ in MMO would be inadvisable: it is trivial to find collisions. Specifically, let H be MMO built on $\widetilde{E}$ with $h_0 = 0^n$. Then $H(a \parallel \widetilde{E}(0, a) \oplus a) = 0^n$ for all $a \in \{0,1\}^n$.

Main Result. Given the recent string of results calling into question the validity of the random-oracle model, it is natural to ask if there are similar results which can be shown for the ideal-cipher model. Specifically, is it possible to exhibit some cryptographic scheme which is provably secure in the ideal-cipher model and yet breaks when instantiated with *any* blockcipher? Given that ideal ciphers are finite objects whereas random oracles are infinite objects, this fact might lead one to ask whether results for the random-oracle model (in particular uninstantiability results) might break down in the ideal-cipher setting given that ideal ciphers can be described with a finite string. We will show that the answer to the above question is "yes": we exhibit a blockcipher-based hash function which is provably collision-resistant in the ideal-cipher model and for which it is trivial to find collisions once the ideal cipher has been instantiated.

We follow the approaches of [5, 15], adapting them to blockciphers and hash functions, and moving into the concrete (rather than asymptotic) setting. The main idea is to create a blockcipher-based hash function $\widetilde{H}$ that acts normally on most inputs, but acts insecurely when given a description of its oracle as an input. In the latter case, $\widetilde{H}$ tests the oracle description embedded in its input against the oracle it already has by submitting some number of test values. If the oracles agree on all values, $\widetilde{H}$ outputs a user-specified value which was also given in the input. The difficulty here is showing that $\widetilde{H}$ remains secure even when behaving this way, and the crucial point is that there a far more possible ideal ciphers with specified input-output pairs than there are encodings to represent them. All of this is formalized and rigorously proven in Section 3.

Related Work. Virtually no discussion of the ideal-cipher model has transpired prior to this work. As already mentioned, much relevant work has appeared in the analogous random-oracle setting. Random oracles were used implicitly at least 18 years ago by Fiat and Shamir in their seminal work on identification schemes [11]. Bellare and Rogaway formalized the notion and argued that the model afforded a path to efficient protocols; as examples, they gave efficient non-malleable and chosen-ciphertext-secure encryption schemes, a signature scheme secure against adaptive chosen-message attack, and an efficient zero-knowledge proof protocol [2]. Canetti, Goldreich, and Halevi gave the first uninstantiable protocol for the random-oracle model: they exhibited a signature scheme which was provably-secure in the random-oracle model but which acted insecurely (gave up its key) when instantiated [5]. Their proof is quite complex, involving techniques similar to Micali's CS-proofs [18]. The same authors later extended their result to show that there exists a signature scheme, limited to short messages, which is also uninstantiable [6]. Maurer, Renner, and Holenstein generalized the results of [5]; they introduced a generalization of indistinguishability called "indifferentiability" which captures the notion of shared random objects (like random oracles) [15]. They state general theorems which imply the result of [5] as a special case, and give an explicit simplified proof of that result. Their proof is very much in the spirit of classical Kolmogorov complexity theory as is ours in the present paper. Nielsen [19] exhibited a protocol that had a simple solution in the random-oracle model, but which had no provable instantiation in the standard model. Bellare, Boldyreva, and Palacio exhibited the first "natural" scheme, a hybrid encryption scheme, secure in the random-oracle model but uninstantiable [1]. Dent adapted techniques from [5] to show an uninstantiable signature scheme in the generic group model (generic groups are finite objects like ideal ciphers) [8].

2 Definitions

Basic notions. Let $\kappa, n \geq 1$ be numbers. A *blockcipher* is a map $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ where, for each $k \in \{0,1\}^\kappa$, the function $E_k(\cdot) = E(k,\cdot)$ is a permutation on $\{0,1\}^n$. Parameter n is called the *blocksize* of E, and n will be understood to be this quantity throughout the paper. If E is a blockcipher then E^{-1} is its inverse, where $E_k^{-1}(y)$ is the string x such that $E_k(x) = y$. Let $\mathrm{Bloc}(\kappa, n)$ be the set of all block ciphers $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. Choosing a random element of $\mathrm{Bloc}(\kappa, n)$ means that for each $k \in \{0,1\}^\kappa$ one chooses a random permutation $E_k(\cdot)$.

A (blockcipher-based) *hash function* is a map $H\colon \mathrm{Bloc}(\kappa, n) \times D \to R$ where $\kappa, n, c \geq 1$, $D \subseteq \{0,1\}^*$, and $R = \{0,1\}^c$. The function H must be given by a program that, given M, computes $H^E(M) = H(E, M)$ using an E-oracle. Hash function $f\colon \mathrm{Bloc}(\kappa, n) \times D \to R$ is a *compression function* if $D = \{0,1\}^a \times \{0,1\}^b$ for some $a, b \geq 1$ where $a + b \geq c$. Fix $h_0 \in \{0,1\}^a$. The *iterated hash* of compression function $f\colon \mathrm{Bloc}(\kappa, n) \times (\{0,1\}^a \times \{0,1\}^b) \to \{0,1\}^a$ is the hash function $H\colon \mathrm{Bloc}(\kappa, n) \times (\{0,1\}^b)^* \to \{0,1\}^a$ defined by $H^E(m_1 \cdots m_\ell) = h_\ell$ where $h_i = f^E(h_{i-1}, m_i)$. Set $H^E(\varepsilon) = h_0$. We often omit the superscript E to f and H.

We write $x \xleftarrow{\$} S$ for the experiment of choosing a random element from the finite set S and calling it x. An *adversary* is an algorithm with access to one or more oracles. We write these as superscripts. The notation $|x|$ denotes the size of the string x, in bits, and the notation $x[i \dots j]$ denotes the substring of string x starting at the i-th bit of x and terminating at the j-th bit, inclusive. All bits are numbered starting from 1, and ascending left-to-right. Finally, $x \,\|\, y$ denotes the concatenation of strings x and y.

Collision resistance. To quantify the collision resistance of a blockcipher-based hash function H we instantiate the blockcipher by a randomly chosen $E \in \mathrm{Bloc}(\kappa, n)$. An adversary A is given oracles for $E(\cdot,\cdot)$ and $E^{-1}(\cdot,\cdot)$ and wants to find a *collision* for H^E—that is, M, M' where $M \neq M'$ but $H^E(M) = H^E(M')$. We look at the number of queries that the adversary makes and compare this with the probability of finding a collision.

Definition 1 (Collision Resistance). Let H be a blockcipher-based hash function, $H\colon \mathrm{Bloc}(\kappa, n) \times D \to R$, and let A be an adversary. Then the advantage of A in finding collisions in H is the real number

$$\mathbf{Adv}_H^{\mathrm{coll}}(A) = \Pr\Big[E \xleftarrow{\$} \mathrm{Bloc}(\kappa, n);\ (M, M') \xleftarrow{\$} A^{E,E^{-1}}\ :\\ M \neq M' \ \wedge\ H^E(M) = H^E(M')\Big]$$

For $q \geq 1$ we write $\mathbf{Adv}_H^{\mathrm{coll}}(q) = \max_A\{\mathbf{Adv}_H^{\mathrm{coll}}(A)\}$ where the maximum is taken over all adversaries that ask at most q oracle queries (ie, E-queries + E^{-1} queries).

3 An Uninstantiable Blockcipher-Based Hash Function

In [3] we find 20 blockcipher-based hash function constructions that are provably secure in the ideal-cipher model. Specifically, it is shown that $\mathbf{Adv}_H^{\mathrm{coll}}(q) = \Theta(q^2/2^n)$ for 20 blockcipher-based hash functions H. This bound is about the best we can hope for: a truly random function would have the same bound due to the birthday phenomenon.

The proofs in [3] are carried out in the ideal-cipher model and the adversaries are information theoretic. In this section we will show that any scheme H from this set can be transformed into a related scheme $\widetilde{H}$ such that $\widetilde{H}$ is uninstantiable. We first outline the method and then give the details.

Main Idea. Our goal is to produce an uninstantiable blockcipher-based hash function. We will do this by transforming some scheme which is provably secure in the ideal-cipher model. For concreteness, select any of the 20 secure schemes from [3] and call it H.

We will describe a related blockcipher-based hash function $\widetilde{H}$ which is uninstantiable. The idea has its roots in Kolmogorov complexity. We adapt the approach of Maurer, Renner, and Holenstein [15]; when $\widetilde{H}$ processes input M, it first decomposes M into three parts: $M = (\pi, c, v)$ where the details of this decomposition are left for later. The first parameter, π is considered to be the encoding of a Universal Turing Machine (UTM), encoded in some well-defined manner. The second parameter $c \in \{0,1\}^\sigma$ is a counter that is ignored by $\widetilde{H}$, and the final parameter $v \in \{0,1\}^n$ is the value that the adversary would like to have output by $\widetilde{H}$.

Now $\widetilde{H}$ uses its blockcipher oracle $\mathcal{O}$ to compute $\mathcal{O}(i, 0^n)$ for all $1 \le i \le |\pi|$. (Why we choose this range will become apparent in the proof below.) It also computes $\pi(i, 0^n)$ for the same set of i-values. If $\mathcal{O}(i, 0^n) = \pi(i, 0^n)$ for all $1 \le i \le |\pi|$, $\widetilde{H}$ outputs v. If not, $\widetilde{H}$ outputs $H(M)$.

Now consider two cases: in the first case, the oracle to $\widetilde{H}$ was an ideal cipher I. This means that it is highly unlikely there is a sufficiently-short Turing-machine encoding, π, such that $\pi(\cdot, 0^n)$ would correctly match I on all $|\pi|$ points, and therefore it is extremely unlikely that we would have $I(i, 0^n) = \pi(i, 0^n)$ for all $1 \le i \le |\pi|$. This means that in all likelihood $\widetilde{H}$ would output $H(M)$, and we know this construction is provably collision resistant. And so in this case $\mathbf{Adv}^{\text{coll}}_{\widetilde{H}}(q)$ is $\Theta(q^2/2^n)$ by [3].

Now consider the case where the oracle to $\widetilde{H}$ is some blockcipher E; in other words we have instantiated oracle $\mathcal{O}$ with blockcipher E. There therefore exists some Turing machine π that implements E. Therefore an adversary may simply output two queries $M_1 = (\pi \parallel 0^\sigma \parallel v)$ and $M_2 = (\pi \parallel 1^\sigma \parallel v)$ for any fixed string $v \in \{0,1\}^n$ he desires. Since $\widetilde{H}$ will discover that $E(i, 0^n) = \pi(i, 0^n)$ for all $1 \le i \le |\pi|$, it will output v for both queries, and this adversary will have trivially found a collision.

Note that things could not be worse for $\widetilde{H}$, in fact: not only can we find collisions, but we can find preimages for any output value, second preimages for any output value, and 2^σ inputs which collide on any chosen value.

A Detailed Description. We now proceed to formalize and prove correct the informal discussion just given. Throughout the remainder of this section, n will denote the blocksize of our blockciphers.

Definition 2. *Blockcipher E is said to be k-efficient if it can be implemented as a Turing machine never requiring more than k steps to produce its output.*

For example, all modern blockciphers are 2^{20}-efficient. For the remainder of this section, k is assumed to be some fixed value. We next exhibit an uninstantiable blockcipher-based hash function. Here, by "uninstantiable" we mean that a given hash function H has $\mathbf{Adv}^{\text{coll}}_{H}(q) = O(q^2/2^n)$, and is therefore secure in the ideal-cipher model, but any instantiation of its blockcipher oracle with a blockcipher E results in a trivially insecure hash function.

For the remainder of this section we will let H denote some blockcipher-based hash function which is known to be secure in the ideal-cipher model (such as MMO, in Figure 1). We now give the algorithm $\widetilde{H}$ which is an uninstantiable variant of H, then we prove its various properties.

```
**Algorithm** $\widetilde{H}(M)$
    **if** $|M| \le n + \sigma$ **then return** $H^{\mathcal{O}}(M)$
    $v \leftarrow M[|M| - n + 1 \ldots |M|]$
    $\pi \leftarrow M[1 \ldots |M| - n - \sigma]$
    **if** $\neg$TuringValid($\pi$) **then return** $H^{\mathcal{O}}(M)$
    **for** $i \leftarrow 1$ **to** $|\pi|$
        Run $\pi$ on input $(i, 0^n)$ for at most $k$ steps
        **if** $\pi$ does not output $n$ bits **then return** $H^{\mathcal{O}}(M)$
        **if** $\pi(i, 0^n) \ne \mathcal{O}(i, 0^n)$ **then return** $H^{\mathcal{O}}(M)$
    **return** $v$
```

Fig. 2. An uninstantiable variant of the provably-secure blockcipher-based hash function H. If the input encodes a valid UTM, we evaluate $|\pi|$ values on this UTM and check against our oracle $\mathcal{O}$. If they match, we simply output v, the last n bits of M. There are σ bits of M which are ignored in order to help the attacker produce 2^{σ} colliding inputs with digest v. The UTM π is run for at most k steps, where k is a fixed parameter of the scheme.

Algorithm $\widetilde{H}$ accepts messages M from the domain $(\{0,1\}^n)^+$ and outputs n bits. As usual, the domain could be extended to $M \in \{0,1\}^*$ with an unambiguous padding rule. We fix two parameters to the algorithm: H, the provably-secure blockcipher-based hash function just mentioned, and a counter-size $\sigma > 0$. We assume the domain of H has been extended to $\{0,1\}^*$ so we can dispense with concerns about message sizes in our construction of $\widetilde{H}$. We further fix some binary encoding scheme for Universal Turing Machines (UTMs) such that any UTM can be encoded into a binary string. Furthermore, we assume there is an efficient function TuringValid that returns `true` when given a string π that is a valid UTM encoding under our fixed convention. Finally, we let $\mathcal{O}$ denote the blockcipher-oracle which is used by $\widetilde{H}$. The algorithm to compute $\widetilde{H}^{\mathcal{O}}(M)$ is given in Figure 2.

We are now faced with arguing that $\widetilde{H}$ is uninstantiable. First notice that $\widetilde{H}$ is efficient: we assume that oracle calls are constant-time, so therefore $H^{\mathcal{O}}$ runs in time linear in the length of the input M. Since we run π for at most k steps, the whole algorithm runs in time $O(k|\pi|) = O(|M|)$.

Theorem 1. [**$\widetilde{H}$ is uninstantiable**] Fix some provably-secure blockcipher-based hash function H and some $\sigma > 0$. Let $\mathcal{O}$ be a blockcipher oracle. Then function $\widetilde{H}$ as described above is uninstantiable.

Proof. There are two things we must prove: first, that $\widetilde{H}$ is secure in the ideal-cipher model. That is, $\mathbf{Adv}^{\mathrm{coll}}_{\widetilde{H}^{\mathcal{O}}}(q) = O(q^2/2^n)$. Second, that $\widetilde{H}^E$ is insecure for any efficient blockcipher E.

We begin by showing $\widetilde{H}^{\mathcal{O}}$ is secure when $\mathcal{O}$ is modeled by an ideal cipher. Fix q and suppose adversary A makes q oracle queries to $\mathcal{O}$. (Throughout the proof, we will assume $q \le 2^{n/2}$ since q-values in excess of this render the bound vacuous.) At the end of this process, A must output a pair of distinct messages M and M' in the hope that $\widetilde{H}^{\mathcal{O}}(M) = \widetilde{H}^{\mathcal{O}}(M')$. The probability that he succeeds is the advantage we wish to bound.

There are two types of collisions A may construct given the outputs from his q queries to $\mathcal{O}$. The first collision is event C_1: there exist two distinct messages M_1, M_2 such that they collide under the original hash function (ie, $H(M_1) = H(M_2)$). We have selected H such that $\Pr[\mathsf{C}_1] = O(q^2/2^n)$. The other type of collision A might construct given his q oracle-query outputs results in event C_2 which we describe next.

Extract π as in line 21, and observe the **for** loop at lines 40 through 43. If at any time, π does not output n bits, or if the n bits it does output do not agree with $\mathcal{O}$, we relegate the computation to H. Therefore we are concerned with the condition that π correctly computes the $|\pi|$ values required by the test on line 43. If $\pi(i, 0^n) = \mathcal{O}(i, 0^n)$ for all $1 \leq i \leq |\pi|$, we say π is a "qualifying" program. We define event C_2 as true if there exists a qualifying program with length at most q bits. If C_2 occurs, A will certainly have set v (computed at line 20) to a colliding value, and so we therefore wish to bound $\Pr[\mathsf{C}_2]$.

Adversary A has made q queries to $\mathcal{O}$ and would like to now encode some qualifying program π into M, with $|\pi| \leq q$. To this end, there are two possibilities: (1) A outputs a program π where C_2 is guaranteed because he has queried $\mathcal{O}$ at all points from 1 to $|\pi|$ and there was a qualifying program, or (2) A outputs a program π where there exists some point j with $1 \leq j \leq |\pi|$ that A did not query, yet C_2 occurs by chance. In the second case, $\widetilde{H}$ will ask $\pi(j)$ and the probability over choices of $\mathcal{O}$ that $\pi(j, 0^n) = \mathcal{O}(j, 0^n)$ is $1/2^n$. Therefore in this case $\Pr[\mathsf{C}_2] \leq 1/2^n$.

We therefore concern ourselves with the first case, where C_2 occurs because A has queried $\mathcal{O}(\cdot, 0^n)$ at all points from 1 to $|\pi|$. The encoding scheme is of course fixed a priori. Therefore $\Pr[\mathsf{C}_2]$ is computed over choices of $\mathcal{O}$. Let Q_ℓ be the event that there exists a qualifying program of size ℓ. So $\mathsf{C}_2 = \mathsf{Q}_1 \vee \cdots \vee \mathsf{Q}_q$. For fixed ℓ there are at most 2^ℓ possible Turing-Valid encodings π with $|\pi| = \ell$. We evaluate, at line 43, $\mathcal{O}(i, 0^n)$ for $1 \leq i \leq |\pi|$. Since we are iterating on the key value for $\mathcal{O}$, there is no permutivity, and therefore outputs will be uniform on $\{0,1\}^n$. This means that, for a fixed i, the probability that $\pi(i, 0^n) = \mathcal{O}(i, 0^n)$ is 2^{-n}. The probability this will happen ℓ times is therefore $2^{-n\ell}$, and given there are 2^ℓ possible encodings, we see

$$\Pr_{\mathcal{O}}[\mathsf{Q}_\ell] \;\leq\; 2^\ell/2^{n\ell} \;=\; 1/2^{\ell(n-1)} \;\leq\; 1/2^{n-1}.$$

So the chance of finding a qualifying program within q queries is

$$\Pr_{\mathcal{O}}[C_2] \;=\; \Pr_{\mathcal{O}}[\mathsf{Q}_1 \vee \cdots \vee \mathsf{Q}_q] \;\leq\; \sum_{\ell=1}^{q} 1/2^{n-1} \;=\; \frac{q}{2^{n-1}}.$$

Finally, the chance that A can find any collision in q queries is bounded by $\Pr[\mathsf{C}_1 \vee \mathsf{C}_2] \leq \Pr[\mathsf{C}_1] + \Pr[\mathsf{C}_2] = O(q^2/2^n) + q/2^{n-1} = O(q^2/2^n)$, as required.

The second case is quite straightforward. We wish to show that $\widetilde{H}^E$ is insecure for any efficient blockcipher E. Since E *does* have a concise Turing-Valid encoding π, we may simply write two messages

$$M = \pi \parallel 0^\sigma \parallel 0^n \qquad \text{and} \qquad M' = \pi \parallel 1^\sigma \parallel 0^n .$$

Since the oracle to $\widetilde{H}$ is E, and since π agrees with E on *every* point, the **if** condition in line 43 will never hold and we will return $v = 0^n$ for each message. Thus we have $\widetilde{H}^E(M) = \widetilde{H}^E(M') = 0^n$, yielding a collision with zero oracle queries required.

Preimage, Second Preimage, and Multicollisions. Since the instantiated form of $\widetilde{H}$ allows us full control over the output, we can clearly find 2^σ preimages for any digest of our choice, and we can similarly find $2^\sigma - 1$ second preimages for any given value. Similarly, we can find multicollisions for any value, and 2^σ collisions for each of the 2^n possible outputs. In this sense, $\widetilde{H}^E$ is much worse than just failing to be collision resistant: it fails to have any security properties at all.

On a technical note, the alert reader will notice that we at no time defined what "insecure" means for a blockcipher-based hash function that has been instantiated. This is because all concrete hash functions are "insecure" if security requires the nonexistence of any efficient program that outputs a colliding pair of inputs! (Since collisions must exist for any non-injective map f, there exists a program that simply outputs a colliding pair for any given f.) Nonetheless, there exists an intuitive notion of security for fixed functions like SHA-1, and clearly the instantiated version of hash function $\widetilde{H}$ is insecure in this sense.

Artificiality. Like all other uninstantiable schemes, $\widetilde{H}$ is quite artificial. It is uninstantiable only because it was designed to be, and upon inspection no one would use such a scheme. It remains to be seen whether there is a more natural construction (where "natural" is necessarily subjective). Thus far, as in the random-oracle model analog, no scheme proven secure in the ideal-cipher model has been broken after instantiation, unless that was the goal from the start.

4 Conclusion and Open Questions

Although the scheme just presented is quite unnatural, it does arouse suspicion as to the wisdom of blindly using the ideal-cipher model in proofs of security. More evidence to support this suspicion could be provided by showing that H^{AES} is insecure for a hash scheme H from [3] that is provably-secure in the ideal-cipher model. Such an attack would necessarily exploit specific features of AES, but since AES is generally thought to be well-designed, it would add fuel to the fire.

Probably the short-signature results of [6] could be extended to this setting, but a more interesting question is whether there exists a "natural" scheme that is provably-secure in the ideal-cipher model but uninstantiable. Hash functions probably are not the right place to look for these, but there are many other objects whose proofs rely on the ideal-cipher model that might provide settings where natural examples of uninstantiable schemes could be constructed.

Acknowledgements

We would like to thank Zully Ramzan, Phillip Rogaway and Thomas Shrimpton for their comments and suggestions. John Black's work was supported by NSF CAREER-0240000 and a gift from the Boettcher Foundation.

References

1. Bellare, M., Boldyreva, A., and Palacio, A. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Advances in Cryptology – EUROCRYPT '04* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 171–188.
2. Bellare, M., and Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security* (1993), pp. 62–73.
3. Black, J., Rogaway, P., and Shrimpton, T. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Advances in Cryptology – CRYPTO '02* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag.
4. Boneh, D. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium* (1998), vol. 1423 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 48–63.
5. Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. In *Proceedings of the 30th ACM Symposium on the Theory of Computing* (1998), ACM Press, pp. 209–218.
6. Canetti, R., Goldreich, O., and Halevi, S. On the random-oracle methodology as applied to length-restricted signature schemes. In *First Theory of Cryptography Conference* (2004), vol. 2951 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 40–57.
7. Cramer, R., and Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '93* (1998), Lecture Notes in Computer Science, Springer-Verlag, pp. 13–25.
8. Dent, A. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology – ASIACRYPT '02* (2002), vol. 2501 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 100–109.
9. Desai, A. The security of all-or-nothing encryption: Protecting against exhaustive key search. In *Advances in Cryptology – CRYPTO '00* (2000), vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag.
10. Even, S., and Mansour, Y. A construction of a cipher from a single pseudorandom permutation. In *Advances in Cryptology – ASIACRYPT '91* (1992), vol. 739 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 210–224.

11. Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86* (1986), Lecture Notes in Computer Science, Springer-Verlag, pp. 186–194.
12. Jaulmes, E., Joux, A., and Valette, F. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In *Fast Software Encryption (FSE 2002)* (2002), vol. 2365 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 237–251.
13. Kilian, J., and Rogaway, P. How to protect DES against exhaustive key search (An analysis of DESX). *Journal of Cryptology 14*, 1 (2001), 17–35.
14. Matyas, S., Meyer, C., and Oseas, J. Generating strong one-way functions with cryptographic algorithms. *IBM Technical Disclosure Bulletin 27*, 10a (1985), 5658–5659.
15. Maurer, U. M., Renner, R., and Holenstein, C. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *First Theory of Cryptography Conference* (2004), vol. 2951 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 21–39.
16. Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography.* CRC Press, 1996.
17. Merkle, R. One way hash functions and DES. In *Advances in Cryptology – CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer-Verlag.
18. Micali, S. CS-proofs. In *Proceedings of IEEE Foundations of Computing* (1994), pp. 436–453.
19. Nielsen, J. B. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO '02* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 111–126.
20. NIST. Secure hash standard (FIPS 180-1). 1995.
21. Pointcheval, D., and Stern, J. Security proofs for signature schemes. In *Advances in Cryptology – EUROCRYPT '96* (1996), Lecture Notes in Computer Science, Springer-Verlag, pp. 387–398.
22. Preneel, B., Govaerts, R., and Vandewalle, J. Hash functions based on block ciphers: A synthetic approach. In *Advances in Cryptology – CRYPTO '93* (1994), Lecture Notes in Computer Science, Springer-Verlag, pp. 368–378.
23. Rabin, M. Digitalized signatures. In *Foundations of Secure Computation* (1978), R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, Eds., Academic Press, pp. 155–168.
24. Schnorr, C.-P. Efficient signature generation by smart cards. *Journal of Cryptology 4*, 3 (1991), 161–174.
25. Shannon, C. Communication theory of secrecy systems. *Bell Systems Technical Journal 28*, 4 (1949), 656–715.
26. Simon, D. Finding collsions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98* (1998), Lecture Notes in Computer Science, Springer-Verlag, pp. 334–345.
27. Winternitz, R. A secure one-way hash function built from DES. In *Proceedings of the IEEE Symposium on Information Security and Privacy* (1984), IEEE Press, pp. 88–90.