Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information

Ran Canetti

IBM T.J. Watson Research Center. Email: *canetti@watson.ibm.com*

Abstract. The *random oracle model* is a very convenient setting for designing cryptographic protocols. In this idealized model all parties have access to a common, public random function, called a *random oracle.* Protocols in this model are often very simple and efficient; also the analysis is often clearer. However, we do not have a general mechanism for transforming protocols that are secure in the random oracle model into protocols that are secure in real life. In fact, we do not even know how to meaningfully *specify* the properties required from such a mechanism. Instead, it is a common practice to simply replace - often without mathematical justification - the random oracle with a 'cryptographic hash function' (e.g., MD5 or SHA). Consequently, the resulting protocols have no meaningful proofs of security.
We propose a research program aimed at rectifying this situation by means of identifying, and subsequently realizing, the useful properties of random oracles. As a first step, we introduce a new primitive that realizes a specific aspect of random oracles. This primitive, called *oracle hashing,* is a hash function that, like random oracles, 'hides all partial information on its input'. A salient property of oracle hashing is that it is probabilistic: different applications to the same input result in different hash values. Still, we maintain the ability to *verify* whether a given hash value was generated from a given input. We describe constructions of oracle hashing, as well as applications where oracle hashing successfully replaces random oracles.

1 Introduction

Existing collision resistant hash functions, such as MD5 [Ri] and SHA [SHA], are very useful and popular cryptographic tools. In particular, these functions (often nicknamed 'cryptographic hash functions') are used in a variety of settings where far stronger properties than collision resistance are required.

Some of these properties are better understood and can be rigorously formulated (e.g., the use as pseudorandom functions [BCK1], or as message authentication functions [BCK2]). Often, however, these extra properties are not precisely specified; even worse, it is often unclear whether the attributed properties can at all be formalized in a meaningful way.

We very roughly sketch two salient such properties. One is 'total secrecy': it is assumed that if h is a cryptographic hash function then $h(x)$ 'gives no information on x'. The other is 'unpredictability': it is assumed to be infeasible to 'find an x such that $x, h(x)$ have some desired property'. This is of course only a sketch; each application requires different variants.

These uses of MD5, SHA, and other cryptographic hash functions are often justified by saying that 'using cryptographic hash functions is equivalent to using random oracles'. More specifically, the following random oracle paradigm is employed. Assume the security of some protocol (that makes use of a 'cryptographic hash function' h) needs to be proven. Then an idealized model is formulated, where there is a public and *random* function R such that everyone can query R on any value x and be answered with $R(x)$. Next modify the protocol so that each invocation of the hash function h is replaced with a query to R. Finally, it is suggested that if the modified construction (using R) is secure in this idealized model then the original construction (using h) is secure in 'real-life'. (We remark that here the random oracle can be viewed as an 'ideal hash function'. In particular, R satisfies both the 'total secrecy' and the 'unpredictability' properties sketched above, since $R(x)$ is a *random number* totally independent of x.)

However, the fact that a construction is secure in the random oracle model does not provide any concrete assurance in the security of this construction in 'real life'. In particular, there exist natural protocols that are secure if a random oracle is used, but are clearly insecure if the random oracle is replaced by *any deterministic function* (and in particular by any cryptographic hash function). The motivating scenario described below provides a good example. (In view of this criticism we stress that, with all its drawbacks, the random oracle model has proved instrumental in designing very useful protocols and applications, as well as new concepts, e.g. [FS, BDMP, M, BR1, BR2, BR3, PS]).

In this work we make a first step towards rigorously specifying some 'random-oracle-like' properties of hash functions. We concentrate on the 'total secrecy' property sketched above. That is, we propose a new primitive, called oracle hashing, that hides all partial information on its input, while maintaining the desired properties of a hash function.

The rest of the introduction is organized as follows. We first sketch a motivating scenario for the new primitive. Next we describe the new primitive, together with several constructions and applications.

A motivating scenario. Consider the following scenario. (It should be kept in mind that, while providing initial intuition for the properties desired from the new primitive, this scenario is of limited scope. In particular, some properties of the new primitive do not come to play here.) Alice intends to publish a puzzle in the local newspaper. She also wants to attach a short string c that will allow readers that solved the puzzle to verify that they have the right solution, but such that c will 'give away' *no partial information* about the solution, x, to readers who have not solved the puzzle themselves. In other words, Alice wants c to mimic an 'ideal scenario' where the readers can call the editor (as many times as they wish), suggest a solution and be answered only 'right' or 'wrong'.

A crypto-practitioner posed with this problem may say: "what's the big deal? c should be a cryptographic hash (e.g., MD5 or SHA) of the solution. It is easily verifiable, and since the hash is one-way c gives no information on the preimage."

Indeed, this ad-hoc solution may be good enough for some practical purposes. However, when trying to 'pin down', or formalize the requirements from a solution some serious difficulties are encountered. In particular, no known cryptographic primitive is adequate. For instance one-way functions are not good enough, since they only guarantee that the *entire* preimage cannot be computed given the function value. It is perfectly possible that a one-way function 'leaks' partial information, say half of the bits of its input.

Furthermore, *any deterministic function* (even ones that hide all the bits of the input, and even 'cryptographic hash functions') are inadequate here, since they are bound to disclose *some* information on the input: For any deterministic function f, $f(x)$ itself is some information on x. One way hash families [NY1] are inadequate for the same reason: they only guarantee that collisions are hard to find, and may leak partial information on the input.

Similarly, commitment schemes (even non-interactive ones) are inadequate since they require the committer to participate in the de-commit stage, whereas here the newspaper editor does not want to be involved in de-committals. (Also, de-committals by nature reveal the correct solution x, even if the suggested solution is wrong.)

In fact, it seems that the only known way to model such a primitive is via the random oracle model: Given access to a random oracle R, Alice can simply publish $c = R(x)$, where x is the solution to the puzzle. This way, given x it can be easily verified whether $c = R(x)$, and as long as the correct x is not found then $R(x)$, being a totally random string, gives no information on x.

We remark that $R(x)$ does in a way provide assistance in finding x since one can now exhaustively search the domain of solutions until a solution x such that $R(x) = c$ is found. This is, however, the same assistance provided by the newspaper in the 'ideal scenario' described above; thus it is a welcome property of a solution.

The new primitive: Oracle Hashing. The proposed primitive, oracle hashing, is designed to replace the random oracle R in the above scenario, as well as in several others. The idea behind this mechanism is quite simple. Traditionally, one thinks of a hash function as a *deterministic* construct, in the sense that two invocations on the same input will yield the same answer. We diverge from this concept, allowing the hash function, H, to be probabilistic in the sense that different invocations on the same input result in different outputs. That is, $H(x)$ is now a random variable depending on the random choices of H. It is this randomization that allows us to require that $H(x)$ will 'hide all partial information on x'.

Oracle hashing also diverges from the notion of (universal, or even one way) hash families [CW, NY1], since there it is usually the case that a *deterministic* function is randomly chosen 'beforehand', and then fixed for the duration of the application.

But now we may have lost the ability to verify hashes. So we require that there exists a verification algorithm, V, that correctly decides, given x and c, whether c a hash of x. (Using standard deterministic hashing, the verification procedure is simple: apply the hash function to x and check whether the result equals c. Here a different procedure may be required.)

This mechanism is somewhat reminiscent of signature schemes, where H takes the role of the signing algorithm and V takes the role of the signature verification algorithm. It is stressed, however, that here no secret keys are involved and both functions can be invoked by everyone. (Also, here additional properties will be required from the pair H, V.)

It remains to formulate the 'secrecy' requirement. This proves to be a non-trivial task. We want to capture the property that 'the hashed value gives no information on the input'. The natural concept that comes to mind is semantic security (originally used for encryption schemes [GM]): *'whatever can be computed given $H(x)$ can also be computed without it'*. But semantic security is unachievable in our scenario, since given $H(x)$ and some value y one must be able to tell whether $x = y$. In particular, if the input x has only a small number of possible values (say 0 or 1) then it is easy to find x from $H(x)$ by running the verification algorithm on all possible inputs.

We thus introduce a new, weaker notion of secrecy, which we call oracle security. This notion essentially means that the only way in which $c = H(x)$ can be used to find information on x is by exhaustively trying different z's and checking if $V(z, c)$ accepts. Very roughly, this can be formulated as follows: Let I_x be the oracle that answers 1 to a query z iff $z = x$; Otherwise it answers 0. Then, *"finding information on x given $H(x)$ is equivalent to finding information on x given only access to the oracle I_x"*. Thus, oracle security is valuable only if there is 'enough uncertainty' about the input, i.e. if no single input is too probable.

We present several equivalent formalizations of oracle security. Furthermore, as in the case of encryption, it is convenient to incorporate in the formalization the notion of 'a-priori information' on the secret value. However here (in contrast with the case of encryption) we don't know whether oracle security without a-priori information is equivalent to oracle security with a-priori information. We elaborate within.

On the constructions. We present a simple oracle hashing scheme based on number-theoretic primitives. Assume a large safe prime p is known. (p is safe if $q = (p-1)/2$ is a prime.) Then, given input x, choose a random element r in Z_p^* and let $H(x) = r^2, r^{2 \cdot h(x)}$, where the calculations are made modulo p, and h is any collision resistant hash function. Verification is straightforward (i.e., to verify whether a pair a, b is a hash of a known message x, check whether $a^{h(x)} \equiv b \pmod{p}$). Here the only requirement from the hash function h is collision resistance. The security of this construction is shown based on strong variants of the Diffie-Hellman assumption. (Different assumptions are needed to show different levels of security.) These assumptions may well be of independent interest.

The above scheme is somewhat costly, since it involves a modular exponentiation. We thus suggest simple constructions based on a cryptographic hash function h. (For instance, let $H(x, r) = r, h(r, h(x))$.) Here we of course make stronger assumptions on h than just collision resistance. We stress however that, in contrast to the 'random oracle heuristic', these are well-defined assumptions.

We remark that constructs similar to the ones described here are implicit in several previous works, sometimes for related purposes (e.g., [F, P, E]). None of these works, however, suggests any primitive similar to the one proposed here. Also, a similar idea is used in the BSD UNIX password file, where a random 'salt' is prepended to a password before encrypting it, and then stored together with the ciphertext.

Applications. A first, immediate application is for scenarios like the 'puzzle in the newspaper' scenario (i.e., whenever one wants to make public a verifiable hash that leaks no information on the hashed value.) Oracle hashing can also be used to replace the use of random oracles in known constructions. We demonstrate this on an encryption function introduced by Bellare and Rogaway [BR1]. This function was proven semantically secure only in the random oracle model described above. (It is suggested in [BR1], as a rule-of-thumb, to replace the random oracle with a cryptographic hash function.) We show that if one replaces oracle hashing for random oracles then the construction becomes secure without resorting to random oracles.

Another application is for content-concealing signatures: Assume that one wants to sign a message m and at the same time make sure that the signature itself hides all partial information on m (from parties who do not already know m). Then, given a message m one can simply sign $H(m)$ instead of signing m. See more details within.

Further research. This work can be viewed as a first step in a research program whose goal is to better understand the random oracle model. This model 'blends' in it several potentially unrelated desired characteristics hash functions, in a way that makes it hard to distinguish which property is being used at each application. Such properties are 'total secrecy' together with several quite different flavors of 'unpredictability'. As demonstrated here, some applications need only some properties and not others. Is it possible to identify additional such properties, and subsequently to realize them without resorting to random oracles?

2 Defining oracle hashing

The definition of oracle hashing consists of three requirements: Completeness and Correctness (that together comprise a validity requirement), together with Secrecy. The first two requirements are fairly standard. Formulating the Secrecy requirement, however, is non-trivial. We present several variants and briefly discuss their relations.

Completeness. This requirement is straightforward: "*Algorithm V will accept (except perhaps with negligible probability) pairs x, c where c is generated by applying H to x.*"

Correctness. We would like to require that: "*It is infeasible to cheat V into accepting pairs x, c such that c was not generated by applying H to x.*" Formalizing this requirement is somewhat tricky since the fact that H is probabilistic make the statement 'c was not generated as $H(x)$' ambiguous. In particular, this requirement takes different flavors depending on whether the producer of the hash is trusted to use H as specified (in which case one only needs to protect against non-malicious errors) or untrusted (in which case one need to protect against malicious efforts to generate ambiguous hashes). We get around these problems by making the stronger requirement that it is infeasible to find 'collisions', i.e. two different inputs x, y and a hash value c such that V accepts c as a legal hash of both x and y.

Secrecy (oracle security). We want to say that: "*Having $c = H(x)$ gives no information on x, besides the ability to exhaustively search the domain for*

x such that $c = H(x)$." This requirement takes different flavors, depending on which probability distributions on the inputs are considered, and on whether the attacker is assumed to have some a-priori information on x. We start with the case where no a-priori information on x is known. Here we present our chosen formalization, together with two other formalizations. We show that all three are equivalent.[1] We believe that comparing the different formalizations helps understanding the nature of oracle security. In particular, two of the formalizations are reminiscent of the two equivalent formalizations of the security of encryption functions (see [G]).

We first need the following definitions: Say that a function $f : \mathbf{N} \to \mathbf{R}$ is negligible if it approaches zero faster than any polynomial (when its input grows to infinity).

Definition 1. Let $\mathcal{X} = \{X_k\}_{k \in \mathbf{N}}$ and $\mathcal{Y} = \{Y_k\}_{k \in \mathbf{N}}$ be two ensembles of probability distributions. We say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable (and write $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$) if for any polytime distinguisher D the difference $|\text{Prob}(D(x) = 1) - \text{Prob}(D(y) = 1)|$ is a negligible function of k, where x is drawn from X_k and y is drawn from Y_k.

Definition 2. A distribution ensemble $\mathcal{X} = \{X_k\}_{k \in \mathbf{N}}$ is well-spread if for any polynomial $p(\cdot)$ and all large enough k, the largest probability of an element in X_k is smaller than $p(k)$ (i.e., $\max_a(X_k(a)) < p(k)$).
(In other words, the max-entropy of distributions in $\mathcal{X}$ must vanish super-logarithmically, see [CG]).

We proceed to the (basic) definition of oracle hashing. Consider a pair of algorithms H, V. Algorithm H, given a security parameter k and input x, chooses a random value in domain R_k and outputs a value c. Algorithm V, given k and input c, outputs a binary value. In the sequel the security parameter, k, is often implicit in our notation.

Definition 3. Say that H, V are an oracle hashing scheme if the following requirements hold.

1. **Completeness:** For all large enough k, for all input x and for $r \in_R R_k$ we have that $\text{Prob}(V(x, H(x, r)) \neq 1)$ is negligible (in k).[2]
2. **Correctness:** For any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ outputs, on input k, a triplet c, x, y such that $x \neq y$ and $V(x, c) = V(y, c) = 1$ is negligible.[3]

[1] Here and for the rest of the discussion we assume non-uniform adversaries. I.e., an adversary is a family of circuits with polynomial size.

[2] $x \in_R D$ means that element x is independently and uniformly chosen from domain D.

[3] Note that in the case that such triplets c, x, y exist, a non-uniform adversary can have a fixed triplet 'wired in' its circuit for each value of k. Thus, it appears to make no sense to require that it is hard to find such triplets. We get around this problem by letting H, V be chosen a-priori from a family of functions, and requiring that any fixed triplet forms a collision only for a small fraction of the functions in the family. See [D].

3. **Secrecy:** For any poly time adversary $\mathcal{A}$ *with binary output,* and any well-spread distribution $\{X_k\}$:

$$\langle x, \mathcal{A}(H(x,r))\rangle \overset{c}{\approx} \langle x, \mathcal{A}(H(y,r))\rangle \tag{1}$$

where $r \in_R R_k$, and x, y are independently drawn from X_k.

Remarks: 1. The Secrecy requirement can be relaxed by taking into account only the uniform distribution on the inputs. We call this variant oracle hashing for random inputs.

2. It appears that limiting $\mathcal{A}$'s output to a binary value is essential for the Secrecy requirement to make sense. In particular, if $\mathcal{A}$ could have arbitrary length output then it could simply output its input, thus making distinguishing between the two sides of (1) easy.

We present two other formalizations of the secrecy requirement (i.e., of oracle security). A somewhat simplified sketch follows.

First is the formalization sketched in the Introduction. We call it Oracle simulatability: Let I_x be the oracle that answers 1 to a query z iff $z = x$; Otherwise it answers 0. Then, *"For any algorithm C' that has access to hashes of x, there exists an algorithm C that has access only to I_x, such that for any distribution on the x's, and any predicate P, C' does not predicts $P(x)$ substantially better than C."*

Second is a formalization reminiscent of security by indistinguishability of encryption functions. We call it Oracle indistinguishability: *For any distinguisher D there exists a set L of polynomially many inputs, such that for any $x, y \notin L$ we have that D distinguishes between hashes of x and hashes of y only with negligible probability."*

We preferred the formalization of Definition 3 since it naturally supports consideration of only specific distributions on the inputs, and since it extends easily to a reasonable definition for the case where a-priori information on the input is known (see Definition 6).

Theorem 4. *The following requirements are equivalent to the* Secrecy *requirement of Definition 3:*

3a. Oracle simulatability: *For any polytime adversary C' and any polynomial $p(\cdot)$ there exists a polytime adversary C, such that for any distribution ensemble $\{X_k\}$, for any polytime predicate $P(\cdot)$, and for all large enough k:*

$$\text{Prob}(C'(H(x,r)) = P(x)) - \text{Prob}(C^{I_x}() = P(x)) < \frac{1}{p(k)}$$

where $r \in_R R_k$, and x is drawn from X_k.

3b. Oracle indistinguishability: *For any polytime distinguisher $\mathcal{D}$ and any polynomial $p(\cdot)$ there exists a polynomial-size family $\{L_k\}$ of sets such that for all large enough k and for all $x, y \notin L_k$:*

$$\text{Prob}(D(H(x,r)) = 1) - \text{Prob}(D(H(y,r)) = 1) < \frac{1}{p(k)}$$

where $r \in_R R_k$.

Proof. See [C].

Oracle security with a-priori information. The secrecy requirement of Definition 3 assumes that no a-priori information on x is known. We formulate a definition requiring that the hashed value gives no *extra* information on the input x, even when some partial information is already known on x. This definition will be needed for the application described in Section 4.

A first attempt to incorporate a-priori information functions in oracle security may be: "*For any algorithm $\mathcal{A}$ and any* a-priori information function *f, we have that $\langle x, \mathcal{A}(f(x), H(x,r))\rangle$ and $\langle x, \mathcal{A}(f(x), H(y,r))\rangle$ are computationally indistinguishable, where r, x, y are chosen at random from their domains.*" This requirement doesn't make sense, though, since $f(x)$ may 'leak' x in full (for instance it may be that $f(x) = x$), in which case $\mathcal{A}$ can use v to verify whether its second input is a hash of x.

But a-priori information functions f that leak all information on their inputs seem uninteresting here: why try to hide x from adversaries that already know it via $f(x)$? We therefore restrict our attention to functions f that do *not* give full information on x (i.e., functions where x can be computed from $f(x)$ only with negligible probability.) We call such functions uninvertible. Note that one-way functions are uninvertible; yet uninvertible functions are a much broader class than one-way functions. For instance, the null function $\forall x, f(x) = \emptyset$ is uninvertible but not one-way.[4] Furthermore, we allow uninvertible functions to be *probabilistic*, (i.e., the function value can be a random variable depending on internal random choices of f). See also the discussion in [GL].

Definition 5. A (probabilistic) function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is uninvertible with respect to distribution ensemble $\{X_k\}$ if for any probabilistic polynomial time algorithm $\mathcal{A}$ and for x taken from X_k, the probability $\text{Prob}(\mathcal{A}(1^k, f(x)) = x)$ is negligible in k, where the probability is taken over the choices of f, $\mathcal{A}$ and x. (We let $\mathcal{A}$ have input 1^k to signify that it may run in time polynomial in k.) When no distribution is specified, uninvertibility with respect to the uniform distribution is implied.

Definition 6. Say that H, V are a strong oracle hash scheme if the Secrecy requirement of Definition 3 is replaced with:

3. **Strong Secrecy (oracle security with a-priori information):** For any algorithm $\mathcal{A}$ *with binary output,* for any well-spread distribution ensemble $\{X_k\}$, and and for any function f that is uninvertible for $\{X_k\}$:

$$\langle x, \mathcal{A}(f(x), H(x,r))\rangle \overset{c}{\approx} \langle x, \mathcal{A}(f(x), H(y,r))\rangle,$$

where $r \in_R R_k$, and x, y are independently drawn from X_k.

Remarks: 1. As in the case of Definition 3, the strong secrecy requirement can be relaxed by taking into account only the uniform distribution on the inputs. We call this variant strong oracle hashing for random inputs. In particular, this variant will suffice for the first application of Section 4.

[4] One-way functions require that it is infeasible to find *any* value in the preimage of $f(x)$.

3 Constructions

We describe some constructions of oracle hash. First we describe a construction based on a number theoretic assumption. Next we describe constructions based on cryptographic hash functions (such as MD5, SHA).

3.1 The r, r^x construction

The construction proceeds as follows. Let p be a large safe prime, i.e. $p = \alpha q + 1$ where α is a small integer (for simplicity we assume $\alpha = 2$). Let Q be the subgroup of order q in Z_p^* (i.e., Q is the group of squares modulo p). On input m and random input $r \in_R Q$, the oracle hash function H first computes $x = h(m)$ where h is a collision resistant hash function; next it outputs $H(m, r) = r, r^x$. (Here and in the sequel calculations are made modulo p.) The verification algorithm V is straightforward: given an input m and a hashed value $\langle a, b\rangle$, compute $x = h(m)$ and accept if $a^x = b$.

We analyze this construction based on three strong variants of the Diffie-Hellman assumption. The variants are used to show, respectively, that the construction satisfies oracle security with random inputs, oracle security, and oracle security with a-priori information. (These notions are defined in Section 2.)

Assumption 7 The Diffie-Hellman Indistinguishability Assumptions: Let k be a security parameter. Let $p = 2q + 1$ be a randomly chosen k-bit safe prime and let $g \in_R Q$ (where Q is the group of squares modulo p).

DHI Assumption I: Let $a, b, c \in_R Z_q^*$. Then, $\langle g^a, g^b, g^{ab}\rangle \overset{c}{\approx} \langle g^a, g^b, g^c\rangle$.
DHI Assumption II: For any well-spread distribution ensemble $\{X_q\}$ where the domain of X_q is Z_q^*, for a drawn from X_q and for $b, c \in_R Z_q^*$ we have $\langle g^a, g^b, g^{ab}\rangle \overset{c}{\approx} \langle g^a, g^b, g^c\rangle$.
DHI Assumption III: For any uninvertible function f and for $a, b, c \in_R Z_q^*$ we have $\langle f(a), g^b, g^{ab}\rangle \overset{c}{\approx} \langle f(a), g^b, g^c\rangle$.

Remarks: 1. It can be seen that Assumption III implies Assumption II, and that Assumption II implies Assumption I. We were unable to show implications in the other direction.

2. While these assumptions are considerably stronger than the standard Diffie-Hellman assumption (there it is only assumed that g^{ab} cannot be computed given p, g, g^a, g^b), they seem consistent with the current knowledge on the Diffie-Hellman problem. In particular, Assumption I appeared in the past, both explicitly and implicitly. It is not hard to see that it is *equivalent* to the semantic security of the El-Gamal encryption scheme [E]. Furthermore, the value exchanged via the DH key exchange is often assumed to be indistinguishable from random. An assumption equivalent to Assumption I is formulated in [B]. Also, this assumption underlies a new and efficient construction of pseudorandom functions [NR].

Although Assumptions II and III look quite strong, we were unable to contradict them. We propose the viability of these assumptions as an open question.

To gain assurance in the plausibility of these assumptions, we remark that it is a common practice to use Diffie-Hellman key exchange modulo a large prime of, say, 1024 bits, but to choose the secret exponents a and b as random numbers of only, say, 200 bits. It is then assumed that the resulting secret, g^{ab}, still has the full '100 bits of security'.[5] This practice implicitly relies on Assumption II (or, alternatively, III) for the case where the first 824 bits of a are fixed to 0.

3. Choosing a safe prime (and the restriction to the subgroup Q) is a standard procedure aimed at avoiding attacks based on the residuocity of a, b, c relative to small factors of $p-1$. It also carries the advantage that any non-zero member of Q is a generator of Q.

4. Naor and Reingold show that if Assumption I is broken then it is possible to distinguish g^a, g^b, g^{ab} from g^a, g^b, g^c for *any* $a, b, c \in Z_q^*$ [NR].

For the analysis of the construction, we first consider a somewhat simplified version, where the collision resistant hash function h is omitted and the input is assumed to be taken from Z_q^*.

Theorem 8. *1. If DHI Assumption I holds then the function $H(x,r) = r, r^x$, together with its verification algorithm, are an oracle hashing scheme for random inputs.*

2. If DHI Assumption II holds then the function $H(x,r) = r, r^x$, together with its verification algorithm, are an oracle hashing scheme.

3. If DHI Assumption III holds then the function $H(x,r) = r, r^x$, together with its verification algorithm, are a strong oracle hashing scheme.

Proof. See [C].

The construction $H(m,r) = r, r^{h(m)}$**.** Strictly speaking, this construction does not satisfy our requirements since the functions h we have in mind are fixed, non-scalable constructs with no asymptotic behavior. Assume however, for sake of the following discussion, that h now describes a scalable collision resistant function where the probability of finding collisions is negligible in the security parameter. (In the next subsection we deal with the non-scalability of existing cryptographic hash functions in a more rigorous way.)

We examine compliance with Definition 3. Completeness still holds. Correctness is now based on the collision resistance of h. (I.e., if two inputs $m \neq m'$ and a hash value c are found such that $V(m,c) = V(m',c) = 1$, then $h(m) = h(m')$.) For the Secrecy requirement, note that as long as the input m is drawn from a well-spread distribution, the value $x = h(m)$ must also be well-spread (otherwise h-collisions may be found by straightforward sampling). Thus, as long as h is collision-resistant, Definition 3 is satisfied under DHI Assumption II; Definition 6 is satisfied under DHI Assumption III.

3.2 Constructions based on cryptographic hash functions

The construction described in the previous subsection is somewhat inefficient since it involves a modular exponentiation. In light of the efficiency of existing

[5] There are several ways to find discrete logarithms of $2k$ bit numbers in $O(2^k)$ steps, regardless of the size of the modulus. See details in [MOV].

cryptographic hash functions (such as MD5 and SHA), and of the general "diffusion and confusion" properties they seem to possess, it is natural to look for a construction based only on such functions. Here making additional new assumptions on these functions is unavoidable. However, in contrast with the 'random oracle heuristic' discussed in the introduction, these will be well defined assumptions.

We propose three simple constructions of oracle hashing, incorporating randomness in the input of the hash function. Each construction (or, mode of operation of the hash function) results in a different assumption on the underlying hash function. The assumption will simply be that using the hash function in the corresponding mode satisfies either Definition 3 or 6, respectively. We let further research and practical experience indicate which construction (if any) is preferable in terms of performance and security.

A first construction that comes to mind, given a cryptographic hash function h is $H(x,r) = r, h(r,x)$, where r is a random string of length β. (Setting $\beta = 128$ for MD5 and $\beta = 160$ for SHA seems appropriate.) Verification (and the Completeness property) are straightforward. Correctness follows directly from the collision resistance of h. The Secrecy requirement imposes the following requirement on h. Following the concrete (i.e., non-asymptotic) security approach of [BKR, BGR, BCK1] we say that:

Definition 9. A hash function h is (τ, δ)-secure with respect to the $H(x,r) = r, h(r,x)$ construction and some distribution Δ on $\{0,1\}^*$ if for any adversary $\mathcal{A}$ and distinguisher D, each running in time τ, we have

$$|\mathrm{Prob}(D(x, \mathcal{A}(r, h(r,x))) = 1) - \mathrm{Prob}(D(x, \mathcal{A}(r, h(r,y))) = 1)| \leq \delta$$

where x, y are independently drawn from Δ and $r \in_{\mathrm{R}} \{0,1\}^\beta$.

(This assumption is obtained by simply plugging the construction in the Secrecy requirement of Definition 3.) A seemingly equivalent variant is $H(x,r) = r, h(r \oplus x)$, where $\oplus$ denotes bitwise exclusive or.

We remark that the "bit commitment scheme based on one way functions" described in [S], p. 87, is secure under the assumption that the one-way function in use satisfies Definition 9. In fact, this assumption seems *necessary* here.

Another possible construction is $H(x,r) = r, h(r, h(x))$. Completeness and correctness are as above. The resulting security assumption can be formulated analogously to the former one. Note that potentially this construction is 'more secure' than the former one, in the sense that if the latter construction fails then most probably the former one fails, but not necessarily vice-versa.

Yet another construction is based on the HMAC construction [BCK2]: let $H(x,r) = r, h(r_1, h(r_2, x))$, where $r_1 = r \oplus \mathrm{opad}$, $r_2 = r \oplus \mathrm{ipad}$, and opad and ipad are two fixed constants. Also here, Completeness and correctness are as above. This construction may be even 'more secure', again in the sense that if the HMAC construction fails then most probably so does the previous one, but not necessarily vice versa.

We remark that embedding the randomness in the IV may result in inferior constructions, since it may simplify violating Correctness. That is, let $h_r(x)$

denote the value of $h(x)$ when the IV is set to r. Consider the construction $H(x,r) = h_r(x)$. Now in order to violate Correctness it suffices to find r, r', x, x' such that $h_r(x) = h_{r'}(x')$. This is a much easier task; See [BB, MOV] for more details.

4 Applications

We describe two more applications, on top of the one described in the Introduction.

Avoiding random oracles. In a sequence of papers Bellare and Rogaway demonstrate how to construct, in the random oracle model, simple, efficient, and provably secure encryption and signature schemes, based on any trapdoor permutation (e.g., the RSA permutation) [BR1, BR2, BR3, BR4]. It is suggested as a 'rule-of-thumb' to replace, in practice, the random oracle with a cryptographic hash function. While the resulting constructions are very attractive and useful in practice, they lack rigorous proofs of security.

It is thus natural to attempt the following procedure with respect to these schemes: (a) replace the random oracle with oracle hashing, and (b) prove the security of the resulting schemes *without random oracles*. We do that to a simple encryption scheme described in [BR1].

The scheme proceeds as follows given a random oracle R, and a trapdoor permutation generator G that on input 1^k outputs a pair f, f^{-1} (where f is a one way permutation and f^{-1} is the inverse of f). The public encryption key is f and the private decryption key is f^{-1}. Given message m and a random input s, let the encryption be $E(m,s) = f(s), R(s) \oplus m$. Decryption is straightforward.

It is shown there that this scheme is semantically secure (in the random oracle model). There, semantic security means that for any two messages m_0, m_1, no polytime adversary $\mathcal{A}$ (with access to the encryption algorithm E and to R) can distinguish between encryptions of m_0 and encryptions of m_1 with more than negligible probability.

We show how to replace R with an oracle hashing scheme H. First however we need to make the following two technical assumptions on H. The first assumption is that the random input r appears explicitly in the output of $H(x,r)$. All the schemes described in this paper have this property. We call such schemes public randomness schemes and write $H(x,r) = r, \tilde{H}(x,r)$.

Let B_k denote the domain of hashes with security parameter k. The second assumption is that there is an 'easy to compute' encoding from B_k to $\{0,1\}^{l(k)}$ for some 'reasonable' length function $l(k)$. The encoding should make sure that when a hash is chosen at random from B_k then the encoded value is distributed (close to) uniform in $\{0,1\}^{l(k)}$. Again, the schemes described in this paper have this property: For the r, r^x scheme, one can use a standard encoding of Z_p^* in, say, $\{0,1\}^{|p|-1}$. For the schemes based on cryptographic hash functions no encoding seems to be needed.

We suggest the following encryption scheme. Given message m and random input r, s compute:

$$E(m,r,s) = f(s), r, \tilde{H}(s,r) \oplus m \tag{2}$$

Again, decryption is straightforward.

Proving semantic security of this construction, based on the fact that H is a strong oracle hash function for random inputs, is quite straightforward. In fact, we use only a considerably weaker secrecy property than the one in Definition 6, namely that $\langle f(x), h(x,r)\rangle \overset{c}{\approx} \langle f(x), h(y,r)\rangle$ where x, y, r are uniformly distributed in their domains.

Theorem 10. *The encryption scheme described in (2) is semantically secure, if H is a strong oracle hash function for random inputs with the additional technical properties described above.*

Proof (sketch): Assume an adversary $\mathcal{A}$ such that $\text{Prob}(\mathcal{A}(E(m_1,s)) = 1) - \text{Prob}(\mathcal{A}(E(m_0,s)) = 1) > \delta$ for some m_0, m_1 and δ. Let p_0 (resp., p_1) denote the probability that $\mathcal{A}$ outputs '1' if it is given $E(m_0,s)$ (resp., $E(m_1,s)$), and let p_* denote the probability that $\mathcal{A}$ outputs '1' given $E(m,s)$, where m is uniformly distributed in its domain. Clearly either $|p_* - p_0| \geq \delta/2$, or $|p_* - p_1| \geq \delta/2$. Assume that $|p_* - p_0| \geq \delta/2$.

Construct a distinguisher D between $\langle f(s), H(s,r)\rangle$ and $\langle f(s), H(s',r)\rangle$, where $s.s', r$ are randomly chosen. (Note that the function f is uninvertible.) Recall that here $H(s,r) = r, \tilde{H}(s,r)$, and that for uniformly chosen s, r the value $H(s,r)$ is uniform in $\{0,1\}^l$ for some l. Given $f(s), r, \xi$ (where ξ is either $\tilde{H}(s,r)$ or $\tilde{H}(s',r)$), D will hand $\mathcal{A}$ the 'ciphertext' $f(s), r, \xi \oplus m_0$. Now, if $\mathcal{A}$ outputs 'm_0' then D outputs '$\xi = \tilde{H}(s,r)$'; otherwise it outputs '$\xi = \tilde{H}(s',r)$'.

Analyzing D is straightforward. (It distinguishes with probability $\delta/2$.) It should only be noted that if $\xi = \tilde{h}(s',r)$ then $\mathcal{A}$ is given an encryption of a uniformly chosen message. □

Content-Concealing signatures. Assume that one wants to sign a document m in a way that if m is known then the signature can be verified as usual, and at the same time make sure that the signature itself hides all partial information on m from parties who do not already know m. We call a signature scheme that has this property content-concealing. Such signatures may become handy, for instance, when the document to be signed has been agreed by the parties in a private way, but the signature has to be broadcasted on a public channel where encryption is unavailable or costly. Another possible scenario is when the signer wants to publish beforehand a signature on a document (say, the quarterly earnings of IBM) but make the document public only at a later date.

As in the 'puzzle in the newspaper' problem, to crypto practitioners it may seem that this problem is already solved: Since cryptographic hash functions are assumed to 'hide all partial information on the input', and since the first step in any digital signature algorithm is to apply a cryptographic hash function to the document, then existing digital signatures are already content-concealing.

Also here, however, this is an illusion. No known (until now) cryptographic primitive solves this problem. Furthermore, also here there is a simple solution in the random oracle model: in the presence of a random oracle R one can simply sign $R(m)$ instead of signing m.

When formalizing the requirement that the signature 'hides all partial information on the input' and at the same time allows for verification, one ends up with the same notion of oracle security used for oracle hash. That is:

Definition 11. A signature scheme is (Strong) content-concealing if, in addition to being a signature scheme (as defined in, say, [GMR]), the signing algorithm satisfies the Secrecy requirement of Definition 3 (resp., 6).

Once content-concealing signatures are defined, a solution is straightforward: *To sign a message m, sign $c = H(m, r)$ (and attach c to the signature), where H, V are an oracle hash scheme and r is randomly chosen.* For verification, first verify the signature on c; next verify that c is a hash of m using the verification algorithm V.

Acknowledgments. First and special thanks are due to Hugo Krawczyk and Oded Goldreich, who spent considerable time trying to make sense of my rambling thoughts and early drafts. In particular, the idea to use the r, r^x construction is Hugo's, and some of the formalizations of oracle security are Oded's.

I thank Shafi Goldwasser for discussions on content-concealing signatures (and Juan Garay and Tal Rabin for suggesting the name). I also thank Rosario Gennaro, Moni Naor and Omer Reingold for very helpful discussions, and Dan Boneh for drawing my attention to [B].

References

[AS] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley, 1992.

[BCK1] M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security", *37th FOCS*, 1996.

[BCK2] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication", *CRYPTO'96*, 1996.

[BGR] M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," *CRYPTO'95*, 1995.

[BKR] M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining." *CRYPTO'94*, 1994.

[BR1] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security*, 62-73, 1993.

[BR2] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption", *EUROCRYPT '94 (LNCS 950)*, 92-111, 1995.

[BR3] M. Bellare and P. Rogaway, "The exact security of digital signatures — How to sign with RSA and Rabin", *EUROCRYPT '96 (LNCS 1070)*, 1996.

[BR4] M. Bellare and P. Rogaway, 'Minimizing the use of random oracles in P1363 encryption schemes", Contribution on IEEE P1364. November 10, 1996.

[BDMP] M. Blum, A. De Santis, S. Micali and G. Persiano, "Non-interactive zero-knowledge", *SIAM Journal on Computing, 20(6):1084-1118*, December 1991.

[BB] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5", *EUROCRYPT'93*, 293-304, 1994.

[B] S. Brands, "An efficient off-line electronic cash system based on the representation problem", CWI TR CS-R9323, 1993.

[C] R. Canetti, "Towards realizing random oracles: Hash functions that hide all partial information", in Theory of Cryptology Library, No. 97-07. `http://theory.lcs.mit.edu/ tcryptol`, 1997.

[CW] J. L. Carter and M. N. Wegman, " Universal classes of hash functions", *JCSS No. 18*, 143-154, 1979.

[CG] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity", *SIAM J. Comp., Vol. 17, No. 2*, 230-261, 1988.

[D] I.B. Damgård, "Collision free hash functions and public key signature schemes", *EUROCRYPT 87 (LNCS 304)*, pp. 203–216, 1988.

[DDN] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography", *23rd STOC*, 1991.

[E] T. El-Gamal, *"Cryptography and logarithms over finite fields"*, Ph.D. Thesis, Stanford University, 1984.

[F] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing", *28th FOCS*, 427-437, 1987.

[FS] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", *CRYPTO'86 (LNCS 263)*, 186-194, 1986.

[G] O. Goldreich, *"Foundations of Cryptography (Fragments of a book)"*, Weizmann Inst. of Science, 1995. (Avaliable at `http://theory.lcs.mit.edu/~cryptol/`)

[GM] Shafi Goldwasser and Silvio Micali, "Probabilistic encryption", *JCSS*, Vol. 28, No 2, 270-299, April 1984.

[GL] O. Goldreich and L. Levin, A Hard-Core Predicate to any One-Way Function, *21st STOC*, 1989, pp. 25-32.

[GMR] S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal of Computing*, 17(2):281–308, April 1988.

[MOV] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, "Handbook of applied cryptography", CRC Press, 1997.

[M] S. Micali, "CS proofs", *35th FOCS*, 436-453, 1994.

[NR] M. Naor and O. Reingold, "The Brain can Compute Pseudo-Random Functions, or Efficient Cryptographic Primitives Based on the Decisional Diffie-Hellman Assumption", manuscript.

[NY1] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications", *21st STOC*, 33-43, 1989.

[NY2] M. Naor and M. Yung, "Public key cryptosystems provably secure against chosen ciphertext attacks", *22nd STOC*, 427-437, 1990.

[P] T. P. Pedersen, "Distributed provers with applications to undeniable signatures", *EUROCRYPT'91*, 1991.

[PS] D. Pointcheval and J. Stern, "Security proofs for signature schemes", *Eurocrypt '96 (LNCS 1070)*, pp. 387-398, 1996.

[RS] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", *CRYPTO'91, (LNCS 576)*, 1991.

[Ri] R. Rivest, "The MD5 message-digest algorithm," IETF Network Working Group, RFC 1321, April 1992.

[S] B. Schneier, *"Applied cryptography", 2nd edition*, Wiley and sons, 1996.

[SHA] FIPS 180, "Secure Hash Standard", Federal Information Processing Standard (FIPS), Publication 180, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., May 1993.