Sound Computational Interpretation of Symbolic Hashes in the Standard Model

Flavio D. Garcia and Peter van Rossum

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands.
{flaviog,petervr}@cs.ru.nl

Abstract. This paper provides one more step towards bridging the gap between the formal and computational approaches to the verification of cryptographic protocols. We extend the well-known Abadi-Rogaway logic with probabilistic hashes and we give a precise semantic interpretation to it using Canetti's oracle hashes. These are probabilistic polynomial-time hashes that hide all partial information. Finally, we show that this interpretation is computationally sound.

1 Introduction

The analysis of security protocols is being carried out mainly by means of two different techniques. On the one hand, from a logical perspective, messages are seen as algebraic objects, generated by some grammar from elementary objects such as keys, nonces, and constants. Cryptographic operations are seen as algebraic operations which are unbreakable. Attackers are typically modelled as so-called Dolev-Yao attackers [DY83], having total control over the network, having no computational limitations, and being only (but absolutely) incapable of breaking cryptographic operations. These logical methods are appealing, because they are relatively easy to use and capture most mistakes commonly made in security protocols.

On the other hand, from a complexity-theory perspective, messages are seen as bit strings and cryptographic operations as functions on bit strings satisfying certain security properties [Gol01]. An attacker here is a resource bounded probabilistic algorithm, limited by running time and/or memory, but capable of breaking cryptographic operations, if that is computationally feasible. The complexity based methods are more general and more realistic, but also more complex.

In the last few years much research has been done to relate these two perspectives [AR02,AJ01,MW04,Her05]. Such a relation takes the form of a function mapping algebraic messages m to (distributions over) bit strings $[\![m]\!]$. This map should relate messages that are observationally equivalent in the algebraic world (meaning that a Dolev-Yao attacker can see no difference between them) to indistinguishable distributions over bit strings (meaning that a computationally bounded adversary can only with negligible probability distinguish the distributions). Such a map allows one to use algebraic methods, possibly even automated, to reason about security properties of protocols and have those reasonings be valid also in the computational world.

The work carried out in the literature on relating these two perspectives mainly deals with symmetric encryption [AR02,MW04] and public key encryption [Her05]. Micciancio and Warinschi [MW04] briefly but explicitly question if this logical approach can be extended to, among other things, collision resistant hashes. Backes, Pfitzmann, and Waidner [BPW06] show that in their simulatability framework [PW00] a sound interpretation of hashes cannot exist, but that it is possible to give a sound interpretation of formal hashes in the simulatability framework using random oracles.

The problem with hashes is that in the algebraic world $h(m)$ and $h(m')$ are indistinguishable for a Dolev-Yao attacker if the attacker does not know m and m'. In the computational world, however, the normal security definition — it must be computationally infeasible to compute any pre-image of a hash value or a hash collision [RS04] — does not guarantee that the hash function hides all partial information about the message; hence there is no guarantee that $[\![h(m)]\!]$ and $[\![h(m')]\!]$ are computationally indistinguishable. A possible solution to this can be found in the work of Canetti and others [Can97a,CMR98] on perfectly one-way functions (a.k.a. oracle hashing). These are computable probabilistic hash functions that hide all partial information of their input (see Section 3.3 for a definition and an example).

Our contribution. We propose an extension to the commonly used Abadi-Rogaway logic of algebraic messages introducing a *probabilistic hash operator* $\mathrm{h}^r(m)$ in the logic, next to the probabilistic symmetric encryption operator $\{\!|m|\!\}_k^r$. Just as the original logic introduces a $\Box$-operator to put in place of undecryptable ciphertext (for us $\Box^r$, since we also deal with repetitions of ciphertexts), we introduce a $\boxtimes^r$-operator to put in place of the hash of an unknown message. In the computational world, we interpret h as a perfectly one-way function and prove that the resulting interpretation is sound.

It is relatively easy to see that the interpretation of messages like $\langle m, \mathrm{h}^r(n, 0)\rangle$ and $\langle m, \mathrm{h}^r(n, 1)\rangle$ are computationally indistinguishable whenever the adversary can not learn n from m. If, however, the adversary can learn n from m, then the messages are not observationally equivalent. The main technical difficulty that has to be overcome is that the adversary can learn part of the argument of the hash from the context, as for example in the message $\langle k, \mathrm{h}^r(n, k)\rangle$.

Overview. Section 2 introduces the message algebra, including the probabilistic encryption and probabilistic hash operators. It also defines the observational equivalence relation on messages. Section 3 then introduces the computational world, giving the security definitions for encryption and hashes. In Section 4 the semantic interpretation $[\![-]\!]$ is defined and Section 5 proves the soundness of this interpretation. Finally, Section 6 discusses further research directions.

2 The algebraic setting

This section describes the message space and the observational equivalence extending the well-known Abadi-Rogaway logic [AR02] of algebraic messages with hashes. These messages are used to describe cryptographic protocols and the observational equivalence tells whether or not two protocol runs are indistinguishable for a global eavesdropper. Here a protocol run is simply the concatenation of all the messages exchanged in the run.

Definition 2.1. Key is an infinite set of *key symbols*, Nonce an infinite set of *nonce symbols*, Const a finite set of *constant symbols*, and Random an infinite set of *randomness labels*. Keys are denoted by $k, k', \ldots$, nonces by $n, n', \ldots$, constants by $c, c', \ldots$, and randomness labels by $r, r', \ldots$. There is one special key called $k_\square$ and for every randomness label r there is a special nonce called $n^r_\boxtimes$. Using these building blocks, *messages* are constructed using algebraic encryption, hashing, and pairing operations:

$$\mathrm{Msg} \ni m := c \mid k \mid n \mid \{\!\!\{m\}\!\!\}^r_k \mid \mathrm{h}^r(m) \mid \langle m, m\rangle \mid \square^r \mid \boxtimes^r .$$

Here k and n do not range over all keys/nonces, but only over the non-special ones. Special symbols ($\square^r$ and $\boxtimes^r$) are used to indicate undecryptable ciphertexts or hash values of unknown messages. When interpreting messages as (ensembles of distributions over) bit strings, we will treat $\square^r$ as if it were $\{\!\!\{0\}\!\!\}^r_{k_\square}$ and $\boxtimes^r$ as if it were $\mathrm{h}^r(n^r_\boxtimes)$.

A message of the form $\{\!\!\{m\}\!\!\}^r_k$ is called an *encryption* and the set of all such messages is denoted by Enc. Similarly, messages of the form $\mathrm{h}^r(m)$ are called *hash values* and the set of all these messages is denoted by Hash. Finally Box denotes the set of all messages of the form $\square^r$ or $\boxtimes^r$. The set of all messages that involve a “random choice” at their “top level”, i.e., $\mathrm{Key} \cup \mathrm{Nonce} \cup \mathrm{Enc} \cup \mathrm{Hash} \cup \mathrm{Box}$, is denoted by RanMsg.

The *closure* of a set U of messages is the set of all messages that can be constructed from U using tupling, detupling, and decryption. It represents the information an adversary could deduce knowing U.

Definition 2.2 (Closure). Let U be a set of messages. The *closure* of U, denoted by $\overline{U}$, is the smallest set of messages satisfying:

1. $\mathrm{Const} \subseteq \overline{U}$;
2. $U \subseteq \overline{U}$;
3. $m, m' \in \overline{U} \implies \langle m, m'\rangle \in \overline{U}$;
4. $\{\!\!\{m\}\!\!\}^r_k, k \in \overline{U} \implies m \in \overline{U}$;
5. $\langle m, m'\rangle \in \overline{U} \implies m, m' \in \overline{U}$.

For the singleton set $\{m\}$, we write $\overline{m}$ instead of $\overline{\{m\}}$.

We define the function *encpat*: $\mathrm{Msg} \to \mathrm{Msg}$ as in Abadi-Rogaway [AR02] which takes a message m and reduces it to a pattern. Intuitively, this is the pattern that an attacker sees in a message given that he knows the messages in U. This function does not replace hashes. Formally, it is defined as follows:

$$encpat(m) = encpat(m, \overline{m})$$

where

$$\begin{aligned}
encpat(\langle m_1, m_2\rangle, U) &= \langle encpat(m_1, U), encpat(m_2, U)\rangle \\
encpat(\{\!|m|\!\}_k^r, U) &= \begin{cases} \{\!|encpat(m, U)|\!\}_k^r, & \text{if } k \in U; \\ \square^{\mathcal{R}(\{\!|m|\!\}_k^r)}, & \text{otherwise.} \end{cases} \\
encpat(\mathrm{h}^r(m), U) &= \mathrm{h}^r(encpat(m, U)) \\
encpat(m, U) &= m \quad \text{in any other case.}
\end{aligned}$$

Here $\mathcal{R}\colon \mathrm{Enc} \cup \mathrm{Hash} \hookrightarrow \mathrm{Random}$ is an injective function that takes an encryption or a hash value and outputs a tag that identifies its randomness. We need this tagging function to make sure that the function *encpat* is injective. That is, we need to make sure that distinct undecryptable messages get replaced by distinct boxes and similarly for *hashpat* below.

Now we define the function *hashpat*: Msg $\to$ Msg which takes a message m and reduces all hashes of unknown (not in U) sub-messages, to $\boxtimes$. This function does not replace encryptions. Formally:

$$hashpat(m) = hashpat(m, \overline{m})$$

where

$$\begin{aligned}
hashpat(\langle m_1, m_2\rangle, U) &= \langle hashpat(m_1, U), hashpat(m_2, U)\rangle \\
hashpat(\{\!|m|\!\}_k^r, U) &= \{\!|hashpat(m, U)|\!\}_k^r \\
hashpat(\mathrm{h}^r(m), U) &= \begin{cases} \mathrm{h}^r(hashpat(m, U)), & \text{if } m \in U; \\ \boxtimes^{\mathcal{R}(\mathrm{h}^r(m))}, & \text{otherwise.} \end{cases} \\
hashpat(m, U) &= m \quad \text{in any other case.}
\end{aligned}$$

Naturally, we now define *pattern* as $pattern = encpat \circ hashpat$.

Example 2.3. Consider the message

$$m = \langle \{\!|\{\!|1|\!\}_{k'}^{r'}, \mathrm{h}^{\tilde{r}}(n)|\!\}_k^r, \mathrm{h}^{\hat{r}}(k), k\rangle.$$

Then $\quad hashpat(m) = \langle \{\!|\{\!|1|\!\}_{k'}^{r'}, \ \boxtimes^t \ |\!\}_k^r, \mathrm{h}^{\hat{r}}(k), k\rangle, \quad$ because n is not in $\overline{m}$,

and $\quad pattern(m) = \langle \{\!| \ \square^s \ , \ \boxtimes^t \ |\!\}_k^r, \mathrm{h}^{\hat{r}}(k), k\rangle, \quad$ because k' is not in $\overline{m}$,

where $t = \mathcal{R}(\mathrm{h}^{\tilde{r}}(n))$ and $s = \mathcal{R}(\{\!|1|\!\}_{k'}^{r'})$.

Definition 2.4 (Observational equivalence). Two messages m and m' are said to be *observationally equivalent*, notation $m \cong m'$, if there is a type preserving permutation σ of $\mathrm{Key} \cup \mathrm{Nonce} \cup \mathrm{Box}$ such that $pattern(m) = pattern(m')\sigma$. Here $pattern(m')\sigma$ denotes simultaneous substitution of x by $\sigma(x)$ in $pattern(m')$, for all $x \in \mathrm{Key} \cup \mathrm{Nonce} \cup \mathrm{Box}$.

From the original setting in [AR02] we inherit the requirement that messages must be acyclic for the soundness result to hold.

Definition 2.5 (Acyclicity). Let m be a message and k, k' two keys. The key k is said to *encrypt* k' *in* m if m has a sub-message of the form $\{\!|m'|\!\}_k^r$ with k' being a sub-message of m'. A message is said to be *acyclic* if there is no sequence $k_1, k_2, \ldots, k_n, k_{n+1} = k_1$ of keys such that k_i encrypts k_{i+1} in m for all $i \in \{1, \ldots, n\}$.

3 The computational setting

This section gives a brief overview of the concepts used in the complexity theoretic approach to security protocols. Much of this is standard; the reader is referred to [GB01,BDJR97] for a thorough treatment of the basic concepts, to [AR02] for the notion of *type-0 security* for cryptographic schemes (see Section 3.2 below), and to [Can97a] for the notion of *oracle hashing* (see Section 3.3 below).

In the computational world, messages are elements of $\mathrm{Str} := \{0,1\}^*$. Cryptographic algorithms and adversaries are probabilistic polynomial-time algorithms. When analyzing cryptographic primitives, it is customary to consider probabilistic algorithms that take an element in $\mathrm{Param} := \{1\}^*$ as input, whose length scales with the security parameter. By making the security parameter large enough, the system should become arbitrarily hard to break.

This idea is formalized in the security notions of the cryptographic operations. The basic one, which is what is used to define the notion of semantically equivalent messages, is that of *computational indistinguishability* of probability ensembles over Str. Here a *probability ensemble over* Str is a sequence $\{A_\eta\}_{\eta\in\mathbb{N}}$ of probability distributions over Str indexed by the security parameter.

Definition 3.1 (Computational indistinguishability). Two probability ensembles $\{A_\eta\}_\eta$ and $\{B_\eta\}_\eta$ are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm A, for all polynomials p, and for large enough η,

$$\mathbb{P}[x \overset{\$}{\leftarrow} A_\eta; A(1^\eta, x) = 1] - \mathbb{P}[x \overset{\$}{\leftarrow} B_\eta; A(1^\eta, x) = 1] < \frac{1}{p(\eta)}.$$

After a brief interlude on probabilistic polynomial-time algorithms in Section 3.1, we give the formal definition of an encryption scheme and its security notion in Section 3.2 and of oracle hashing in Section 3.3.

3.1 Probabilistic algorithms

In Definition 3.1, the notion of probabilistic polynomial-time algorithm was already used. Because we explicitly use two different views of these algorithms and in order to fix notation, we give a more precise definition.

Definition 3.2. Coins is the set $\{0,1\}^\omega$, the set of all infinite sequences of 0's and 1's. We equip Coins with the probability distribution obtained by flipping a fair coin for each element in the sequence.

Definition 3.3. The result of running a probabilistic algorithm A on an input $x \in \mathrm{Str}$ is a probability distribution $A(x)$ over Str. When we need to explicitly write the randomness used while running A, we write $A(x, \rho)$ with $\rho \in \mathrm{Coins}$. Using this notation, $A(x)$ and $[\rho \xleftarrow{\$} \mathrm{Coins}; A(x, \rho)]$ are the same probability distribution. When confusion is unlikely, we will also denote the support of this probability distribution, $\{y \in \mathrm{Str} | \mathbb{P}[\rho \xleftarrow{\$} \mathrm{Coins}; A(x, \rho = y)] > 0\}$, by $A(x)$.

Now suppose that A runs in polynomial time p. Then running A on x cannot use more than $p(|x|)$ coin flips. Letting $\mathrm{Coins}_{p(|x|)}$ denote the uniform probability distribution on $\{0, 1\}^{p(|x|)}$, we get that the probability distribution $A(x)$ can also be written as $[\rho \xleftarrow{\$} \mathrm{Coins}_{p(|x|)}; A(x, \rho)]$.

3.2 Encryption scheme

For each security parameter $\eta \in \mathbb{N}$ we let $\mathrm{Plaintext}_\eta \subseteq \mathrm{Str}$ be a non-empty set of *plaintexts*, satisfying that for each $\eta \in \mathbb{N}$: $\mathrm{Plaintext}_\eta \subseteq \mathrm{Plaintext}_{\eta+1}$ as in Goldwasser and Bellare [GB01]. Let us define $\mathrm{Plaintext} = \bigcup_\eta \mathrm{Plaintext}_\eta$. There is a set $\mathrm{Keys} \subseteq \mathrm{Str}$ of *keys* and also a set $\mathrm{Ciphertext} \subseteq \mathrm{Str}$ of *ciphertexts*. Furthermore, there is a special bit string $\perp$ not appearing in Plaintext or Ciphertext. An *encryption scheme Π* consists of three algorithms:

1. a (probabilistic) key generation algorithm $\mathcal{K}\colon \mathrm{Param} \to \mathrm{Keys}$ that outputs, given a unary sequence of length η, a randomly chosen element of Keys;
2. a (probabilistic) encryption algorithm $\mathcal{E}\colon \mathrm{Keys} \times \mathrm{Str} \to \mathrm{Ciphertext} \cup \{\perp\}$ that outputs, given a key and a bit string, a possibly randomly chosen element from Ciphertext or $\perp$;
3. a (deterministic) decryption algorithm $\mathcal{D}\colon \mathrm{Keys} \times \mathrm{Str} \to \mathrm{Plaintext} \cup \{\perp\}$ that outputs, given a key and a ciphertext, an element from Plaintext or $\perp$.

These algorithms must satisfy that the decryption (with the correct key) of a ciphertext returns the original plaintext. The element $\perp$ is used to indicate failure of en- or decryption, although there is no requirement that decrypting with the wrong keys yields $\perp$.

Now we define type-0 security of an encryption scheme as in [AR02], which is a variant of the standard semantic security definition, enhanced with some extra properties. In particular a type-0 secure encryption scheme is which-key concealing, repetition concealing and length hiding. We refer to the original paper for motivation and explanations on how to achieve such an encryption scheme. The notion of type-0 security makes slightly unrealistic assumptions on the encryption scheme. However our result on hashes does not significantly depend on the specific security notion for the encryption scheme. As in [MP05,Her05], it is possible to replace type-0 security by the standard notion of ind-cpa or ind-cca by adapting the definition of *encpat*. For simplicity of the exposition, throughout this paper we adopt the former security notion.

Definition 3.4. An *adversary (for type-0 security)* is a probabilistic polynomial-time algorithm $A^{\mathcal{F}(-),\mathcal{G}(-)}\colon \mathrm{Param} \to \{0, 1\}$ having access to two probabilistic oracles $\mathcal{F}, \mathcal{G}\colon \mathrm{Str} \to \mathrm{Str}$. The *advantage* of such an adversary is the function $\mathrm{Adv}_A \colon \mathbb{N} \to \mathbb{R}$ defined by

$$\mathrm{Adv}_A(\eta) = \mathbb{P}[\kappa, \kappa' \xleftarrow{\$} \mathcal{K}(1^\eta); A^{\mathcal{E}(\kappa,-),\mathcal{E}(\kappa',-)}(1^\eta) = 1] - \\ \mathbb{P}[\kappa \xleftarrow{\$} \mathcal{K}(1^\eta); A^{\mathcal{E}(\kappa,0),\mathcal{E}(\kappa,0)}(1^\eta) = 1].$$

Here the probabilities are taken over the choice of κ and κ' by the key generation algorithm, over the choices of the oracles, and over the internal choices of A. An encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is called *type-0 secure* if for all polynomial-time adversaries A as above, the advantage Adv_A is a negligible function of η. This means that for all positive polynomials p and for large enough η, $\mathrm{Adv}_A(\eta) \leq \frac{1}{p(\eta)}$.

In the sequel we need an extra assumption on the encryption scheme, namely that the ciphertexts are well-spread as a function of the coins tosses of $\mathcal{E}$. It means that for *all* plaintexts μ and *all* keys κ, no ciphertext is exceptionally likely to occur as the encryption of μ under κ. Note that this does not follow from, nor implies type-0 security. Also note that every encryption scheme running in cipher block chaining mode automatically has this property: the initial vector provides the required randomness.

Definition 3.5 (Well-spread). An encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is said to be *well-spread* if for every polynomial p,

$$\forall \eta \gg 1 . \forall x \in \mathrm{Ciphertext} . \forall \kappa \in \mathcal{K}(1^\eta) . \forall \mu \in \mathrm{Plaintext}_\eta \colon \; \mathbb{P}[\mathcal{E}(\kappa, \mu) = x] < \frac{1}{p(\eta)}.$$

3.3 Oracle hashing

The underlying secrecy assumptions behind formal or Dolev-Yao hashes [DY83] are very strong. It is assumed that given a hash value $f(x)$, it is not possible for an adversary to learn any information about the pre-image x. In the literature this idealization is often modelled with the random oracle [BR93]. Such a primitive is not computable and therefore it is also an idealization. Practical hash functions like SHA or MD5 are very useful cryptographic primitives even though this functions might leak partial information about their input. Moreover, under the traditional security notions (one-wayness), a function that reveals half of its input is considered secure. In addition, any deterministic hash function f leaks partial information about x, namely $f(x)$. Through this paper we consider a new primitive introduced by Canetti [Can97a] called *oracle hashing*, that mimics what semantic security is for encryption schemes. This hash function is probabilistic and therefore it needs a verification function, just as in a signature scheme. A *hash scheme* consists of two algorithms $\mathcal{H}$ and $\mathcal{V}$. The probabilistic algorithm $\mathcal{H} \colon \mathrm{Param} \times \mathrm{Str} \to \mathrm{Str}$ takes a unary sequence and a message and outputs a hash value; the verification algorithm $\mathcal{V} \colon \mathrm{Str} \times \mathrm{Str} \to \{0, 1\}$ that given two messages x and c correctly decides whether c is a hash of x or not. As an example we reproduce here a hash scheme proposed in the original paper. Let p be a large (i.e., scaling with η) safe prime. Take $\mathcal{H}(x) = \langle r^2, r^{2 \cdot h(x)} \mod p \rangle$, where r is a randomly chosen element in $\mathbb{Z}_p^*$ and h is any collision resistant hash function. The verification algorithm $\mathcal{V}(x, \langle a, b\rangle)$ just checks whether $b = a^{h(x)} \bmod p$.

Canetti gives essentially two security notions for such a hash scheme. The first one, *oracle indistinguishability*, guarantees that an adversary can gain no information at all about a bit string, given its hash value (or rather, with sufficiently small probability). The second one is an appropriate form of collision resistance. It guarantees that an adversary cannot (or rather, again, with sufficiently small probability) compute two distinct messages that successfully pass the verification test with the same hash value.

Definition 3.6. A hash scheme $\langle \mathcal{H}, \mathcal{V}\rangle$ is said to be *oracle indistinguishable* if for every family of probabilistic polynomial-time predicates $\{D_\eta \colon \mathrm{Str} \to \{0,1\}\}_{\eta\in\mathbb{N}}$ and every positive polynomial p there is a polynomial size family $\{L_\eta\}_{\eta\in\mathbb{N}}$ of subsets of Str such that for all large enough η and all $x, y \in \mathrm{Str} \setminus L_\eta$:

$$\mathbb{P}[D_\eta(\mathcal{H}(1^\eta, x)) = 1] - \mathbb{P}[D_\eta(\mathcal{H}(1^\eta, y)) = 1] < \frac{1}{p(\eta)}.$$

Here the probabilities are taken over the choices made by $\mathcal{H}$ and the choices made by D_η. This definition is the non-uniform [Gol01] version of oracle indistinguishability proposed by Canetti [Can97a] as it is finally used throughout the proof (See the full version [Can97b], Appendix B).

Definition 3.7 (Collision resistance). A hash scheme $\langle \mathcal{H}, \mathcal{V}\rangle$ is said to be *collision resistant* if for every probabilistic polynomial time adversary A, the probability

$$\mathbb{P}[\langle c, x, y\rangle \xleftarrow{\$} A(1^\eta); x \neq y \wedge \mathcal{V}(x, c) = \mathcal{V}(y, c) = 1]$$

is a negligible function of η.

4 Interpretation

Section 2 describes a setting where messages are algebraic terms generated by some grammar. In Section 3 messages are bit strings and operations are given by probabilistic algorithms operating on bit strings. This section shows how to map algebraic messages to (distributions over) bit strings. This interpretation is very much standard. We refer to [AR02,AJ01,MW04] for a thorough explanation. In particular this section introduces notation that allows us to assign, beforehand, some of the random coin flips used for the computation of the interpretation of a message. This notation becomes useful throughout the soundness proof.

Definition 4.1. For every message m and set of messages V we define the set $\mathrm{R}(m, V) \subseteq \mathrm{RanMsg}$ of *random messages in m relative to V* as follows: if $m \in V$, then $\mathrm{R}(m, V) = \emptyset$, otherwise

$$\begin{aligned} \mathrm{R}(c, V) &= \emptyset & \mathrm{R}(\{\!|m|\!\}_k^r, V) &= \mathrm{R}(m, V) \cup \{k, \{\!|m|\!\}_k^r\} \\ \mathrm{R}(n, V) &= \{n\} & \mathrm{R}(\mathrm{h}^r(m), V) &= \mathrm{R}(m, V) \cup \{\mathrm{h}^r(m)\} \\ \mathrm{R}(k, V) &= \{k\} & \mathrm{R}(\langle m_1, m_2 \rangle, V) &= \mathrm{R}(m_1, V) \cup \mathrm{R}(m_2, V) \\ \mathrm{R}(\square^r, V) &= \{k_\square, \square^r\} & \mathrm{R}(\boxtimes^r, V) &= \{n_\boxtimes^r, \boxtimes^r\}. \end{aligned}$$

The set of *random messages in m* is defined as $\mathrm{R}(m) := \mathrm{R}(m, \emptyset)$ and the set of *random messages in m relative to m'* as $\mathrm{R}(m, m') := \mathrm{R}(m, \{m'\})$.

Note that $\mathrm{R}(m)$ is nearly equal to the set of all sub-messages of m that are in RanMsg; the only difference is that $\mathrm{R}(m)$ also may contain the special key $k_\square$ or special nonces $n_\boxtimes^r$. When interpreting a message m as (ensembles of distributions over) bit strings (Definition 4.4 below), we will first choose a sequence of coin flips for all elements of $\mathrm{R}(m)$ and use these sequences as source of randomness for the appropriate interpretation algorithms.

Also note that $\mathrm{R}(m, m')$ is the set of all random messages in m except those that *only* occur as a sub-message of m' (see Definition 4.5 below).

Example 4.2. Let m be the message $\langle k, \{\!|0|\!\}_k^r, \mathrm{h}^{r'}(\{\!|0|\!\}_k^r, n), n' \rangle$ and let $\tilde{m}$ be the message inside the hash: $\langle \{\!|0|\!\}_k^r, n \rangle$. Then the randomness in m is $\mathrm{R}(m) = \{k, \{\!|0|\!\}_k^r, \mathrm{h}^{r'}(\{\!|0|\!\}_k^r, n), n', n\}$, the randomness inside the hash is $\mathrm{R}(\tilde{m}) = \{\{\!|0|\!\}_k^r, k, n\}$, and the randomness that occurs only outside the hash is $\mathrm{R}(m, \mathrm{h}^{r'}(\tilde{m})) = \mathrm{R}(m) \setminus \{\mathrm{h}^{r'}(\tilde{m}), n\}$. The randomness that is shared between the inside of the hash and the outside of the hash is $\mathrm{R}(m, \mathrm{h}^{r'}(\tilde{m})) \cap \mathrm{R}(\tilde{m}) = \{\{\!|0|\!\}_k^r\}$.

Definition 4.3. For every finite set X we define $\mathrm{Coins}(X)$ as $\{\tau \colon X \to \mathrm{Coins}\}$. We equip $\mathrm{Coins}(X)$ with the induced product probability distribution. Furthermore, for every message m we write $\mathrm{Coins}(m)$ instead of $\mathrm{Coins}(\mathrm{R}(m))$.

An element of τ of $\mathrm{Coins}(m)$ gives, for every sub-message m' of m that requires random choices when interpreting this sub-message as a bit string, an infinite sequence $\tau(m')$ of coin flips that will be used to resolve the randomness.

Now we are ready to give semantic to our message algebra. We use $\mathcal{E}$ to interpret encryptions, $\mathcal{K}$ to interpret key symbols, and $\mathcal{H}$ to interpret for hashes. We let $\mathcal{C} \colon \mathrm{Const} \to \mathrm{Str}$ be a function that (deterministically) assigns a constant bit string to each constant identifier. We let $\mathcal{N} \colon \mathrm{Param} \to \mathrm{Str}$ be the nonce generation function that, given a unary sequence of length η, chooses uniformly and randomly a bit string from $\{0,1\}^\eta$.

Definition 4.4. For a message m, a value of the security parameter $\eta \in \mathbb{N}$, a finite set U of messages containing $\mathrm{R}(m)$, and for a choice $\tau \in \mathrm{Coins}(U)$ of (at least) all the randomness in m, we can (deterministically) create a bit string

$[\![m]\!]_\eta^\tau \in \mathrm{Str}$ as follows:

$$\begin{aligned}
[\![c]\!]_\eta^\tau &= \mathcal{C}(c) & [\![\{\!|m|\!\}_k^r]\!]_\eta^\tau &= \mathcal{E}([\![k]\!]_\eta^\tau, [\![m]\!]_\eta^\tau, \tau(\{\!|m|\!\}_k^r)) \\
[\![k]\!]_\eta^\tau &= \mathcal{K}(1^\eta, \tau(k)) & [\![\mathrm{h}^r(m)]\!]_\eta^\tau &= \mathcal{H}(1^\eta, [\![m]\!]_\eta^\tau, \tau(\mathrm{h}^r(m))) \\
[\![n]\!]_\eta^\tau &= \mathcal{N}(1^\eta, \tau(n)) & [\![\square^r]\!]_\eta^\tau &= \mathcal{E}([\![k_\square]\!]_\eta^\tau, \mathcal{C}(0), \tau(\square^r)) \\
[\![\langle m_1, m_2\rangle]\!]_\eta^\tau &= [\![m_1]\!]_\eta^\tau [\![m_2]\!]_\eta^\tau & [\![\boxtimes^r]\!]_\eta^\tau &= \mathcal{H}(1^\eta, [\![n_\boxtimes^r]\!]_\eta^\tau, \tau(\boxtimes^r)).
\end{aligned}$$

Note that $[\![m]\!]_\eta^\tau = [\![m]\!]_\eta^{\tau|_{\mathrm{R}(m)}}$. For a fixed message m and $\eta \in \mathbb{N}$, choosing τ from the probability distribution $\mathrm{Coins}(\mathrm{R}(m))$ creates a probability distribution $[\![m]\!]_\eta$ over Str:

$$[\![m]\!]_\eta := [\tau \xleftarrow{\$} \mathrm{Coins}(m); [\![m]\!]_\eta^\tau].$$

Note that although the codomain of $\tau \in \mathrm{Coins}(m)$ is Coins, the set of *infinite* bit strings, when interpreting a fixed message m at a fixed value of the security parameter η, only a predetermined *finite* initial segment of each sequence of coin flips will be used by $\mathcal{K}$, $\mathcal{N}$, $\mathcal{E}$, and $\mathcal{H}$ (cf. Definition 3.3). Denoting by $\mathrm{Coins}_\eta(m)$ the probability distribution (on $\{\tau\colon \mathrm{R}(m) \to \mathrm{Str}\}$) that is actually being used when computing $[\![m]\!]_\eta$, we could also write

$$[\![m]\!]_\eta = [\tau \xleftarrow{\$} \mathrm{Coins}_\eta(m); [\![m]\!]_\eta^\tau].$$

Furthermore, letting η range over $\mathbb{N}$ creates an ensemble of probability distributions $[\![m]\!]$ over Str, namely $[\![m]\!] := \{[\![m]\!]_\eta\}_{\eta\in\mathbb{N}}$.

Definition 4.5. We will also need a way of interpreting a message as a bit string when the interpretation of certain sub-messages has already been chosen in some other way. For this, let e be a function from some set $\mathrm{Dom}(e) \subseteq \mathrm{Pat}$ to Str and let $\tau \in \mathrm{Coins}(U, \mathrm{Dom}(e))$ with U a finite set of messages containing $\mathrm{R}(m)$. We interpret a message m using e whenever possible and τ otherwise: if $m \in \mathrm{Dom}(e)$, then $[\![m]\!]_\eta^{e,\tau} = e(m)$, otherwise

$$\begin{aligned}
[\![c]\!]_\eta^{e,\tau} &= \mathcal{C}(c) & [\![\{\!|m|\!\}_k^r]\!]_\eta^{e,\tau} &= \mathcal{E}([\![k]\!]_\eta^\tau, [\![m]\!]_\eta^{e,\tau}, \tau(\{\!|m|\!\}_k^r)) \\
[\![k]\!]_\eta^{e,\tau} &= \mathcal{K}(1^\eta, \tau(k)) & [\![\mathrm{h}^r(m)]\!]_\eta^{e,\tau} &= \mathcal{H}(1^\eta, [\![m]\!]_\eta^{e,\tau}, \tau(\mathrm{h}^r(m))) \\
[\![n]\!]_\eta^{e,\tau} &= \mathcal{N}(1^\eta, \tau(n)) & [\![\square^r]\!]_\eta^{e,\tau} &= \mathcal{E}([\![k_\square]\!]_\eta^{e,\tau}, \mathcal{C}(0), \tau(\square^r)) \\
[\![\langle m_1, m_2\rangle]\!]_\eta^{e,\tau} &= [\![m_1]\!]_\eta^{e,\tau} [\![m_2]\!]_\eta^{e,\tau} & [\![\boxtimes^r]\!]_\eta^{e,\tau} &= \mathcal{H}(1^\eta, [\![n_\boxtimes^r]\!]_\eta^{e,\tau}, \tau(\boxtimes^r)).
\end{aligned}$$

Definition 4.6. We also need a way of pre-specifying some of the random choices to be made when interpreting a message. For this, let $\tau \in \mathrm{Coins}(U)$ for some finite set of messages U. Then for every $\eta \in \mathbb{N}$ and every message m, the distribution $[\![m]\!]_\eta^\tau$ is obtained by randomly choosing coins for the remaining randomness labels in m. Formally,

$$[\![m]\!]_\eta^\tau := [\tau' \xleftarrow{\$} \mathrm{Coins}(\mathrm{R}(m) \setminus U); [\![m]\!]_\eta^{\tau\cup\tau'}],$$

where $\tau \cup \tau' \in \mathrm{Coins}(m)$ denotes the function which agrees with τ on $U \cap \mathrm{R}(m)$ and with τ' on $\mathrm{R}(m) \setminus U$.

This can also be combined with the previous way of preselecting a part of the interpretation. For a function e from a set $\mathrm{Dom}(e) \subseteq \mathrm{Pat}$ to Str and $\tau \in \mathrm{Coins}(U)$ as above, we define $[\![m]\!]_\eta^{e,\tau} := [\tau' \xleftarrow{\$} \mathrm{Coins}(\mathrm{R}(m) \setminus U); [\![m]\!]_\eta^{e,\tau\cup\tau'}]$.

5 Soundness

This section shows that the interpretation proposed in the previous section is computationally sound. Throughout this section we assume that the encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is type-0 secure (or ind-cca with *encpat* modified as in [Her05,MP05]) and well-spread, and that the probabilistic hash scheme $\langle \mathcal{H}, \mathcal{V} \rangle$ is oracle indistinguishable and collision resistant.

The following lemma uses all these assumptions. It claims that if you prespecify some, but not all, of the sequences of coins to be chosen when interpreting a message m, then no single bit string x is exceptionally likely to occur as the interpretation of m.

Lemma 5.1 *Let m be a message, $U \subsetneq \mathrm{R}(m)$. Let p be a positive polynomial. Then*

$$\forall \eta \gg 1 . \forall \tau \in \mathrm{Coins}(U) . \forall x \in \mathrm{Str} : \mathbb{P}[\alpha \xleftarrow{\$} [\![m]\!]_\eta^\tau ; \alpha = x] < \frac{1}{p(\eta)}.$$

Proof. The proof follows by induction on the structure of m. See the full version of this paper [GR06].

Theorem 5.2 *Let m be a message with a sub-message of the form $\mathrm{h}^r(\tilde{m})$. Assume that $\tilde{m} \notin \overline{m}$. Take $m' := m[\mathrm{h}^r(\tilde{m}) := \boxtimes^s]$, where $s = \mathcal{R}(\mathrm{h}^r(\tilde{m}))$. Then $[\![m]\!] \equiv [\![m']\!]$.*

Proof. Assume that $[\![m]\!] \not\equiv [\![m']\!]$, say $A \colon \mathrm{Param} \times \mathrm{Str} \to \{0,1\}$ is a probabilistic polynomial-time adversary and p a positive polynomial such that

$$\frac{1}{p(\eta)} \leq \mathbb{P}[\mu \xleftarrow{\$} [\![m]\!]_\eta ; A(1^\eta, \mu) = 1] - \mathbb{P}[\mu \xleftarrow{\$} [\![m']\!]_\eta ; A(1^\eta, \mu) = 1] \qquad (1)$$

for infinitely many $\eta \in \mathbb{N}$. We will use this to build a distinguisher as in Definition 3.6 that breaks oracle indistinguishability of $\langle \mathcal{H}, \mathcal{V} \rangle$.

Let $\eta \in \mathbb{N}$, abbreviate $\mathrm{R}(m, \tilde{m}) \cap \mathrm{R}(\tilde{m})$ to U and let $\tau \in \mathrm{Coins}(U)$. Note that τ chooses coin flips for the randomness that occurs both inside and outside the hash. Then define a probabilistic polynomial-time algorithm $D_\eta^\tau \colon \{0,1\}^* \to \{0,1\}$ as follows.

algorithm $D_\eta^\tau(\alpha)$:
  $\mu \xleftarrow{\$} [\![m]\!]_\eta^{\{\mathrm{h}^r(\tilde{m}) \mapsto \alpha\},\tau}$
  $\beta \xleftarrow{\$} A(\eta, \mu)$
  return β

This algorithm tries to guess if a given bit string α was drawn from $[\![\mathrm{h}^r(\tilde{m})]\!]_\eta^\tau$ or from $[\![\boxtimes^s]\!]_\eta^\tau = [\![\mathrm{h}^s(n_\boxtimes^s)]\!]_\eta^\tau$. It does so by computing an interpretation for m as follows. The sub-message $\mathrm{h}^r(\tilde{m})$ is interpreted as α; the randomness that is shared between the inside of the hash ($\tilde{m}$) and the rest of the message is resolved using hard-coded sequences of coin flips τ. It then uses the adversary A to guess if the resulting interpretation was drawn from $[\![m]\!]_\eta$ (in which case it guesses that α was drawn from $[\![\mathrm{h}^r(\tilde{m})]\!]_\eta$) or from $[\![m']\!]_\eta$ (in which case it guesses that α was drawn from $[\![\boxtimes^s]\!]_\eta$).

Even though τ has values in Coins, i.e., infinite strings, this is still a well-defined probabilistic polynomial-time algorithm, as it uses only a finite, predetermined amount of bits from τ (cf. Definitions 3.3 and 4.4). However, $(1^\eta, \alpha) \mapsto D_\eta^\tau(\alpha)$ would not be a well-defined probabilistic polynomial-time algorithm.

Now consider one of the infinitely many values of η for which (1) holds. Using D_η^τ we can rephrase (1) as follows:

$$\begin{aligned}
\frac{1}{p(\eta)} &\leq \mathbb{P}[\tau \xleftarrow{\$} \mathrm{Coins}_\eta(U), \alpha \xleftarrow{\$} [\![\mathrm{h}^r(\tilde{m})]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] - \\
&\qquad \mathbb{P}[\tau \xleftarrow{\$} \mathrm{Coins}_\eta(U), \alpha \xleftarrow{\$} [\![\boxtimes^s]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] \\
&= \sum_{\tau \in \mathrm{Coins}_\eta(U)} \Big(\mathbb{P}[\alpha \xleftarrow{\$} [\![\mathrm{h}^r(\tilde{m})]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] - \\
&\qquad \mathbb{P}[\alpha \xleftarrow{\$} [\![\boxtimes^s]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] \Big) \cdot \mathbb{P}[T \xleftarrow{\$} \mathrm{Coins}_\eta(U); T = \tau] \\
&= \sum_{\tau \in \mathrm{Coins}_\eta(U)} \Big(\mathbb{P}[\alpha \xleftarrow{\$} [\![\tilde{m}]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] - \\
&\qquad \mathbb{P}[\alpha \xleftarrow{\$} [\![n_\boxtimes^s]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] \Big) \cdot \mathbb{P}[T \xleftarrow{\$} \mathrm{Coins}_\eta(U); T = \tau].
\end{aligned}$$

Note that τ selects the randomness that is shared between the inside of the hash and the outside of the hash; when α is drawn from $[\![\tilde{m}]\!]_\eta^\tau$ the randomness that appears only inside the hash is chosen (and the assumption on $\tilde{m}$ means that there is really something to choose); $\mathcal{H}$ chooses the randomness for taking the hash; and D_η^τ itself resolves the randomness that appears only outside the hash.

This means that there must be a particular value of τ, say $\bar{\tau}_\eta$, such that

$$\frac{1}{p(\eta)} \leq \mathbb{P}[\alpha \xleftarrow{\$} [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}; D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] - \mathbb{P}[\alpha \xleftarrow{\$} [\![n_\boxtimes^s]\!]_\eta^{\bar{\tau}_\eta}; D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1]. \quad (2)$$

Gathering all $D_\eta^{\bar{\tau}_\eta}$ together for the various values of η, let D be the non-uniform adversary $\{D_\eta^{\bar{\tau}_\eta}\}_{\eta \in \mathbb{N}}$. Note that we have not actually defined $D_\eta^{\bar{\tau}_\eta}$ for all η, but only for those (infinitely many) for which (1) actually holds. What D does for the other values of η is irrelevant.

We will now show that D breaks the oracle indistinguishability of $\langle \mathcal{H}, \mathcal{V} \rangle$. For this, let $L = \{L_\eta\}_{\eta \in \mathbb{N}}$ be a polynomial size family of subsets of Str. We have to show that for infinitely many values of η, there are $x, y \in \mathrm{Str} \setminus L_\eta$ such that D meaningfully distinguishes between $\mathcal{H}(1^\eta, x)$ and $\mathcal{H}(1^\eta, y)$.

Once again, take one of the infinitely many values of η for which (1) holds. Continuing from (2), a short computation (see the full version of this paper [GR06]) gives

$$\frac{1}{p(\eta)} \leq \frac{1}{2p(\eta)} + \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \Big[\Big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]\Big) \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}=\beta]\Big]. \quad (3)$$

Now suppose that for all $\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ and all $\beta \in [\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ we have

$$\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1] < \frac{1}{2p(\eta)}.$$

Then, continuing from (3), we get a contradiction:

$$\begin{aligned}
\frac{1}{p(\eta)} &< \frac{1}{2p(\eta)} + \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \frac{1}{2p(\eta)}\cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}=\beta] \\
&= \frac{1}{2p(\eta)} + \frac{1}{2p(\eta)} \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta}=\beta] \\
&\leq \frac{1}{2p(\eta)} + \frac{1}{2p(\eta)}\ .
\end{aligned}$$

Therefore, there must be an $x \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ and a $y \in [\![n^s_\boxtimes]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ such that

$$\frac{1}{2p(\eta)} \leq \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,x))=1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,y))=1].$$

Hence D breaks oracle indistinguishability, contradicting the assumption on $\langle\mathcal{H},\mathcal{V}\rangle$. □

Theorem 5.3 (Abadi-Rogaway) *Let m be an acyclic message. Suppose that for every sub-message $\mathrm{h}^r(\tilde{m})$ of m, $\tilde{m} \in \overline{m}$. Then $[\![m]\!] \equiv [\![encpat(m)]\!]$.*

Proof. The proof follows just like in Abadi-Rogaway [AR02]. Interpreting hashes here is straightforward because their argument is always known, by assumption. We refer the reader to the original paper for a full proof. □

Theorem 5.4 (Soundness) *Let m and m' be acyclic messages. Then $m \cong m' \implies [\![m]\!] \equiv [\![m']\!]$.*

Proof. The assumption that $m \cong m'$ means that there is a permutation σ of Key ∪ Nonce ∪ Box such that $pattern(m) = pattern(m')\sigma$. Therefore we get $[\![pattern(m)]\!] \equiv [\![pattern(m')]\!]$. By definition of *pattern*, $[\![encpat \circ hashpat(m)]\!] \equiv [\![encpat \circ hashpat(m')]\!]$. Now, by applying Theorem 5.3 two times, we obtain $[\![hashpat(m)]\!] \equiv [\![hashpat(m')]\!]$. Finally, by repeatedly applying Theorem 5.2 on both sides we get $[\![m]\!] \equiv [\![m']\!]$. □

6 Conclusions and future work

We have proposed an interpretation for formal hashes that is computationally sound. For the proof we considered non-uniform adversaries and the assumption that the encryption scheme is type-0 secure and well-spread and that the hash scheme is oracle indistinguishable and collision resistant. This paper considers passive adversaries. It would be interesting to study whether this result can be extended to active adversaries. Another interesting research direction would be proving completeness for this extended logic.

References

AJ01. Martín Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation. In Naoki Kobayashi and Benjamin C. Pierce, editors, *Proceedings of the Fourth International Symposium on Theoretical Aspects of Computer Software (TACS'01)*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2001.

AR02. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

BDJR97. Mihir Bellare, Anand Desai, Eron Jokipii, and Philip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 394–405. IEEE, 1997.

BPW06. Michael Backes, Birgit Pfitzmann, and Michael Waidner. Limits of the reactive simulatability/UC of Dolev-Yao models with hashes. Cryptology ePrint Archive, Report 2006/014 (http://eprint.iacr.org/2006/068), 2006.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM CCS*, pages 62–73. ACM, 1993.

Can97a. Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burt Kaliski, editor, *Advances in Cryptology, 17th Annual International Cryptology Conference (CRYPTO'97)*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997.

Can97b. Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. Cryptology ePrint Archive, Report 1997/007 (http://eprint.iacr.org/1997/007), 1997.

CMR98. Ran Canetti, Danielle Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 131–140. ACM, 1998.

DY83. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

GB01. Shafi Goldwasser and Mihir Bellare. *Lecture Notes on Cryptography*. 2001. http://www-cse.ucsd.edu/~mihir/papers/gb.html.

Gol01. Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

GR06. Flavio D. Garcia and Peter van Rossum. Sound computational interpretation of formal hashes. Cryptology ePrint Archive, Report 2006/014 (http://eprint.iacr.org/2006/014), 2006.

Her05. Jonathan Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, 340(1):57–81, 2005.

MP05. Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In Joe Kilian, editor, *Theory of Cryptography: Second Theory of Cryptography Conference (TCC'05)*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187. Springer, February 2005.

MW04. Daniele Micciancio and Bogdan Warinschi. Completeness theorems of the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

PW00. Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM CCS*, pages 245–254, 2000.

RS04. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption: 11th International Workshop (FSE'04)*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.