# Foundations of Non-Malleable Hash and One-Way Functions

Alexandra Boldyreva[1] and David Cash[1] and Marc Fischlin[2] and Bogdan Warinschi[3]

[1]Georgia Institute of Technology, USA
{aboldyre,cdc}@cc.gatech.edu

[2]Darmstadt University of Technology, Germany
marc.fischlin@gmail.com

[3] University of Bristol, UK
bogdan.warinschi@gmail.com

**Abstract**

Non-malleability is an interesting and useful property which ensures that a cryptographic protocol preserves the independence of the underlying values: given for example an encryption $\mathcal{E}(m)$ of some unknown message $m$, it should be hard to transform this ciphertext into some encryption $\mathcal{E}(m^*)$ of a related message $m^*$. This notion has been studied extensively for primitives like encryption, commitments and zero-knowledge. Non-malleability of one-way functions and hash functions has surfaced as a crucial property in several recent results, but it has not undergone a comprehensive treatment so far. In this paper we initiate the study of such non-malleable functions. We start with the design of an appropriate security definition. We then show that non-malleability for hash and one-way functions can be achieved, via a theoretical construction that uses perfectly one-way hash functions and simulation-sound non-interactive zero-knowledge proofs of knowledge (NIZKPoK). We also discuss the complexity of non-malleable hash and one-way functions. Specifically, we show that such functions imply perfect one-wayness and we give a black-box based separation of non-malleable functions from one-way permutations (which our construction bypasses due to the "non-black-box" NIZKPoK). We exemplify the usefulness of our definition in cryptographic applications by showing that non-malleability is necessary and sufficient to securely replace one of the two random oracles in the IND-CCA encryption scheme by Bellare and Rogaway, and to improve the security of client-server puzzles.

**Keywords:** Non-malleability, hash function, perfect one-wayness.

# 1 Introduction

MOTIVATION. Informally, non-malleability of some function $f$ is a cryptographic property that asks that learning $f(x)$ for some $x$ does not facilitate the task of generating some $f(x^*)$ so that $x^*$ is related to $x$ in some non-trivial way. This notion is especially useful when $f$ is used to build higher-level multi-user protocols where non-malleability of the protocol itself is crucial (e.g., for voting or auctioning). Non-malleability has been rather extensively studied for some cryptographic primitives. For example, both definitions as well as constructions from standard cryptographic assumptions are known for encryption, commitments and zero-knowledge [15, 5, 27, 14, 18, 12, 1, 13, 25, 26, 2]. Non-malleability in the case of other primitives, notably for one-way functions and for hash functions,[1] has only recently surfaced as a crucial property in several works [6, 7, 10, 17], which we discuss below.

For instance, plenty of cryptographic schemes are only proved secure in the random oracle (RO) model [4], where one assumes that a hash function behaves as a truly random function to which every party has access to. It is well-known that such proofs do not strictly guarantee security for instantiations with hash functions whose only design principles are based on one-wayness and/or collision-resistance, because random functions posses multiple properties the proofs may rely on. Hiding all partial information about pre-images, i.e. perfect one-wayness, is one of these properties, and has been studied in [8, 11]. Non-malleability is another example of such a property.

An illustrative example is the encryption scheme of Bellare and Rogaway [4], where a ciphertext of message $M$ has the form $(f(r), G(r) \oplus M, H(r, M))$ for a trapdoor permutation $f$, hash functions $G, H$ and random $r$. The scheme is known to be IND-CCA secure in the random oracle model. However, an instantiation of $H$ with a malleable function for which given $H(r, M)$ it is possible to compute $H(r, M \oplus M')$, for some fixed $M'$ known to the attacker, renders the scheme insecure: the attacker can recover $M$ by submitting to the decryption oracle the valid ciphertext $(f(r), G(r) \oplus M \oplus M', H(r, M \oplus M'))$.

It was shown in [6] that a similar attack can be carried out against the popular OAEP encryption scheme whenever the instantiation of the underlying hash function is malleable. A subsequent work [7] showed that some form of non-malleability permits positive results about security of an alleviated version of the OAEP scheme in the standard model. However, it remains unclear if the approach to non-malleability in [7] expands beyond the OAEP example, and the work left open the construction of non-malleable primitives.

Another motivating example is the abstraction used to model hash functions in symbolic (Dolev-Yao) security analysis. In this setting it is *axiomatized* that an adversary can compute some hash only when it knows the underlying value. Clearly, malleable hash functions do not satisfy this axiom. Therefore, non-malleability for hash functions is necessary in order to ensure that symbolic analysis is (in general) sound with respect to the standard cryptographic model. Otherwise, real attacks that use malleability can not be captured/discovered in the more abstract symbolic model.

In a different vein, and from a more conceptual perspective, higher-level protocols could potentially benefit from non-malleable hash functions as a building block. A recent concrete example is the recommended use of such non-malleable hash functions in a human-computer interaction protocol for protecting local storage [10]. There, access should be linked to the ability to answer human-solvable puzzles (similar to CAPTCHAs), but it should be infeasible for a machine to maul puzzles and redirect them under a different domain to other human beings.

We will also discuss a construction of a cryptographic puzzle from [23] designed to prevent DoS attacks, and show that non-malleability of the underlying hash is necessary for its security.

Hence, non-malleability is a useful design principle that designers of new hash functions should keep in mind. At this point, however, it is not even clear what the exact requirements from a theoretical viewpoint are. Therefore, a first necessary step is to find a suitable definition which is (a) achievable, and (b) applicable. The next step would be to design practical hash functions and compression functions which are non-malleable, or which at least satisfy some weaker variant of non-malleability.

[1] In the sequel we aggregate both one-way functions and hash functions under the term hash functions for simplicity.

CONTRIBUTIONS. In this paper we initiate the foundational study of non-malleable hash functions. We start with the design of an appropriate security definition. Our definition uses the standard simulation paradigm, also employed in defining non-malleability for encryption and commitment schemes. It turns out however that a careless adjustment of definitions for other primitives yield definitions for non-malleable hash functions that cannot be realized. We therefore motivate and provide a meaningful variation of the definition which ensure that the notion is achievable and may be useful in applications.

Testifying to the difference to other cryptographic primitives, we note that for non-malleable encryption the original simulation-based definition of [15] was later shown to be equivalent to an indistinguishability-based definition [5]. For our case here, finding an equivalent indistinguishability-based definition for non-malleable hash functions appears to be far from trivial, and we leave the question as an interesting open problem.

We then show that our definition can be met. Our construction of a non-malleable hash function employs a perfectly one-way hash function (POWHF) [8, 11], i.e., a probabilistic hash function which hides all information about its pre-image. Notice that this form of secrecy in itself does not ensure non-malleability, so we make the function non-malleable by appending a simulation-sound non-interactive zero-knowledge proof of knowledge (NIZKPoK) [27, 12] of the hashed value. Both primitives exist, for example, if trapdoor permutations exist.[2]

The construction we provide is probabilistic and does not achieve the desired level of efficiency for practical applications. We emphasize that our construction should be regarded as a feasibility result that shows that, in principle, non-malleable hash functions can be built from standard assumptions. We leave open the problem of finding a practical, deterministic solution. We note that our definition is general enough to allow such constructions.

Next, we investigate necessary cryptographic assumptions for building non-malleable hash functions. We provide two results. First we show that a non-malleable hash function needs to hide any information about the pre-image. This result justifies the use of POWHFs in our construction. Then we show (in the style of Impagliazzo-Rudich [22]) that black-box constructions of non-malleable one-way functions from one-way permutations are in fact impossible; to be more precise, we follow the approach of Hsiao and Reyzin [21] and show that no black-box security reduction is possible. Notice that our construction circumvents the impossibility result due to the use of a "non-black-box" NIZKPoK.

Finally, we study the applicability of our definition. We show that non-malleability is in fact sufficient for secure partial instantiation of the aforementioned encryption scheme of Bellare and Rogaway [4], i.e., that the scheme remains IND-CCA secure when $H$ is replaced with a non-malleable hash function. Although G is still a random oracle, this partial instantiation helps is to better understand the necessary properties of the primitives and also provides a better security heuristic.

We also sketch an application to the framework of cryptographic puzzles [23] as a defense against DoS attacks, where non-malleability surfaces as an important property. The usefulness of the definition has also been shown in [17], using a special case of a preliminary version of our definition to prove that HMAC [3] is a secure message authentication code, assuming that the compression function of the hash function is non-malleable. We expect further applications of non-malleable hash functions in other areas, and some of the techniques used in our proof here may be helpful for these scenarios.

RELATED WORK. Independently of our work, Canetti and Dakdouk [9] and Pandey et al. [24] recently also suggested one-way functions with special properties related to, yet different from non-malleability. The work of Canetti and Dakdouk [9] introduces the notion of extractable perfect one-way functions where generating an image also guarantees that one knows a preimage. This should even hold if an adversary sees related images, somewhat resembling our non-malleability setting. Yet, extractability in [9] is defined by having a knowledge extractor generate a preimage from the adversary's view, including the other images. In contrast, the common (and our) approach to non-malleability is to deny the simulator the other images, in order to capture the idea that these images should not help.

[2] We remark that the intuitively appealing approach of using non-malleable encryption or commitment schemes to directly construct non-malleable hashes does not work. One of the reasons is that the former primitives rely on secret randomness, whereas hash values need to be publicly verifiable given the pre-image.

The work by Pandey et al. [24] defines adaptive one-way functions where inversion for an image under some index $\mathsf{tag}$ is still infeasible, even if one is allowed to obtain preimages under different indices $\mathsf{tag}'$. This notion is also related to non-malleability and turns out to be useful to design non-malleable protocols like commitments and zero-knowledge proofs. Unfortunately, this strong notion is not known to be realizable at this point.

It is noteworthy that, analogously to our work here, both papers choose the Bellare-Rogaway encryption scheme as a test case (among others). This gives three different views on the requirements of the hash functions in this encryption scheme. While we only instantiate one of the two random oracles, the authors of [9] and of [24] aim at instantiating both hash functions, requiring pseudorandomness as another property of the hash function. Yet, neither extractable perfect one-way functions which are also pseudorandom, nor adaptive pseudorandom generators are known to be constructible under common assumptions. In contrast, our single-oracle instantiation through a non-malleable hash function is possible under standard assumptions.

## 2 Preliminaries

If $x$ is a string then $|x|$ denotes its bit length and if $x, y$ are strings then $x\|y$ denotes the concatenation of $x$ and $y$. If $k \in \mathbb{N}$ then $1^k$ is the string consisting of $k$ consecutive "1" bits. If $S$ is a set then we write $x_1, x_2, \ldots \overset{\$}{\leftarrow} S$ for selecting $x_1, x_2, \ldots$ independently at random from $S$. If $\mathcal{X}$ is a distribution then $x \overset{\$}{\leftarrow} \mathcal{X}(1^k)$ stands for a sampling process where $x$ is picked according to $\mathcal{X}$ for input $1^k$. The term "PPT" (resp. "PT") stands for "probabilistic polynomial-time," (resp. "polynomial-time") and "PPTA" (resp. "PTA") for "PPT algorithm" (resp. "PT algorithm"). If $A$ is a PPTA then the notation $y \overset{\$}{\leftarrow} A(x_1, x_2, \ldots)$ denotes that $y$ is assigned the outcome of $A$ running on inputs $x_1, x_2, \ldots$. If $A$ is deterministic (PTA), we often drop the dollar sign above the arrow.

**Definition 2.1 (Hash Functions)** *A hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ consists of PPTAs for key generation, evaluation and verification, where*

- *PPTA $\mathsf{HK}$ for security parameter $1^k$ outputs a key $K$ (which contains $1^k$ and implicitly defines a domain $D_K$),*
- *PPTA $\mathsf{H}$ for inputs $K$ and $x \in D_K$ returns a value $y \in \{0,1\}^*$,*
- *PTA $\mathsf{HVf}$ on inputs $K, x, y$ returns a decision bit.*

*It is required that for any $K \overset{\$}{\leftarrow} \mathsf{HK}(1^k)$, any $x \in D_K$, any $y \overset{\$}{\leftarrow} \mathsf{H}(K, x)$, algorithm $\mathsf{HVf}(K, x, y)$ outputs $1$.*

Note that we consider a very general syntax, comprising the "classical" notions of one-way functions (with a public key) and of collision-resistant hash functions which compress the input to a shorter digest (see [20] for definitions). In our case the evaluation algorithm $\mathsf{H}$ may be probabilistic, as long the correctness of hash values is verifiable given the pre-image only (via $\mathsf{HVf}$). Also, we do not demand the length of the output of the hash function to be smaller than that of the input. However, while we capture a large class of primitives, the generalized syntax may not preserve all properties of the special cases, e.g., if the evaluation algorithm is probabilistic, two independent parties hashing the same input will not necessarily get the same value.

We now use the above syntax to recall the definitions of one-wayness and collision resistance. For one-wayness the definition that we give is more general than the standard one in that it considers specific input distributions $\mathcal{X}$ for the function, and also accounts for the possibility that the adversary may have some partial information about the pre-image (modeled through a probabilistic function $\mathsf{hint}$):

**Definition 2.2 (One-wayness and Collision-resistance)** *A hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ is called*

- one-way *(wrt $\mathcal{X}$ and $\mathsf{hint}$) if for any PPTA $\mathcal{A}$ the probability that for $K \overset{\$}{\leftarrow} \mathsf{HK}(1^k)$, $x \overset{\$}{\leftarrow} \mathcal{X}(1^k)$, $h_x \overset{\$}{\leftarrow} \mathsf{hint}(K, x)$, $y \overset{\$}{\leftarrow} \mathsf{H}(K, x)$ and $x^* \overset{\$}{\leftarrow} \mathcal{A}(K, y, h_x)$ we have $\mathsf{HVf}(K, x^*, y) = 1$, is negligible.*
- collision-resistant *if for any PPTA $\mathcal{A}$ the probability for $K \overset{\$}{\leftarrow} \mathsf{HK}(1^k)$, $(x, x', y) \overset{\$}{\leftarrow} \mathcal{A}(K)$ that $x \neq x'$ but $\mathsf{HVf}(K, x, y) = 1$ and $\mathsf{HVf}(K, x', y) = 1$, is negligible.*

## 3 Non-Malleability of Hash and One-Way Functions

Our definition for hash functions follows the classical (simulation-based) approach for defining non-malleability [15]. Informally, our definition requires that for any adversary which, on input a hash value $y$, finds another value $y^*$ such that the pre-images are related, there exists a simulator which does just as well *without ever seeing* $y$.

In the adversary's attack we consider a three-stage process. The adversary first selects a distribution $\mathcal{X}$ from which a secret input $x$ is then sampled (and passes on some state information st). In the second stage the algorithm sees a hash value $y$ of this input $x$, and the adversary's goal is to create another hash value $y^*$ (usually different from $y$). In the third stage the adversary is given $x$ and now has to output a pre-image $x^*$ to $y^*$ which is "related" to $x$ (we make the definition stronger by giving the challenge pre-image to the adversary). The simulator may also pick a distribution $\mathcal{X}$ according to which $x$ is sampled, but then the simulator needs to specify $x^*$ directly from the key of the hash function only.

In the second stage the adversary (and consequently the simulator) also gets as input a "hint" $h_x$ about the original pre-image $x$, to represent some a-priori information potentially gathered from other executions of other protocols in which $x$ is used. In fact, such side information is often crucial for the deployment in applications, e.g., for the encryption example in Section 6. As in the case of non-malleable commitments and encryption, related pre-images are defined via a relation $R(x, x^*)$. This relation may also depend on the distribution $\mathcal{X}$ to catch significantly diverging choices of the adversary and the simulator and to possibly restrict the choices for $\mathcal{X}$, say, to require a certain min-entropy. However, unlike for other primitives, we do not measure the success of the adversary and the simulator for arbitrary relations $R$ between $x$ and $x^*$, but instead restrict the relations to a class $\mathcal{R}$ of admissible relations. We discuss this and other subtleties after the definition:

**Definition 3.1 (NM-Hash)** *A hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ is called non-malleable (with respect to probabilistic function* $\mathsf{hint}$ *and relation class $\mathcal{R}$)*[3] *if for any PPTA $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$ there exists a PPTA $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_x)$ such that for every relation $R \in \mathcal{R}$ the difference*

$$\Pr\left[\,\mathbf{Exp}^{\textit{nmh-1}}_{\mathcal{H},\mathcal{A}}(k) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\textit{nmh-0}}_{\mathcal{H},\mathcal{S}}(k) = 1\,\right] \quad \text{is negligible, where :}$$

```
*Experiment* $\mathbf{Exp}^{\textit{nmh-1}}_{\mathcal{H},\mathcal{A}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \xleftarrow{\$} \mathcal{A}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k)$, $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$
    $y \xleftarrow{\$} \mathsf{H}(K, x)$
    $(y^*, st_y) \xleftarrow{\$} \mathcal{A}_y(y, h_x, st_d)$
    $x^* \xleftarrow{\$} \mathcal{A}_x(x, st_y)$
    *Return 1 iff*
        $R(\mathcal{X}, x, x^*)$
        $\wedge\, (x, y) \neq (x^*, y^*) \,\wedge\, \mathsf{HVf}(K, x^*, y^*) = 1$
```

```
*Experiment* $\mathbf{Exp}^{\textit{nmh-0}}_{\mathcal{H},\mathcal{S}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \xleftarrow{\$} \mathcal{S}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k)$, $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$


    $x^* \xleftarrow{\$} \mathcal{S}_x(h_x, st_d)$
    *Return 1 iff*
        $R(\mathcal{X}, x, x^*)$
```

REMARK 1. Our definition is parameterized by a class of relations $\mathcal{R}$. This is because for some relations the definition is simply not achievable, as in the case when the relation involves the hash of $x$ instead of $x$ itself. More specifically, consider the relation $R(x, x^*)$ which parses $x^*$ as $K, y$ and outputs $\mathsf{HVf}(K, x, y)$. Then, an adversary on input $y, h_x, \mathrm{st}_d$ may output $y^* \xleftarrow{\$} \mathsf{H}(K, (K, y))$ and then, given $x$, returns $x^* = (K, y)$. This adversary succeeds in experiment $\mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}}(k)$ with probability 1. In contrast, any simulator is likely to fail, as long as the hash function does not have "weak" keys, i.e., keys for which the distribution of generated images is non-trivial (such that the simulator can guess $y$ with sufficiently high probability).

[3] Throughout the paper all hint functions and relations are assumed to be efficient. We furthermore assume that the security parameter is given in unary to all algorithms as additional input (if not mentioned explicitly).

We resolve this problem by requiring the definition to hold for a subset $\mathcal{R}$ of all relations. It is of course desirable to seek secure constructions with respect to very broad classes of relations (cf. our construction in Section 4) which are more handy for general deployment. At the same time, certain scenarios may only require non-malleability with respect to a small set of relations (cf. the application example discussed in Section 6). Our definition is general enough and permits easy tuning for the needs of a particular application or a class of applications.

REMARK 2. For virtually all "interesting" functions $\mathcal{H}$ and relation classes $\mathcal{R}$ the definition is achievable only for adversaries and simulators that output descriptions of well-spread distributions $\mathcal{X}$ (i.e., with super-logarithmic min-entropy) and so-called uninvertible functions $\mathsf{hint}$ [8] (for which finding the exact pre-image is infeasible). Note that uninvertibility is a weaker requirement than one-wayness, as it holds for example for constant functions. We prefer to keep the definition as general as possible, so we do not explicitly impose such restrictions on the adversary, simulator, and $\mathsf{hint}$.

REMARK 3. In our definition we demand that the simulator outputs $x^*$ given $K$ and $h_x$ only. A weaker condition would be to have a simulator $\mathcal{S}_y(h_x, \mathrm{st}_d)$ first output $y^*$, like the adversary $\mathcal{A}_y$, and then $x^* \leftarrow \mathcal{S}_x(x, \mathrm{st}_y)$, before checking that $R(\mathcal{X}, x, x^*)$ and that $\mathsf{HVf}(K, x^*, y^*) = 1$. We call this a *weak simulator* and hash functions achieving this notion *weakly non-malleable*. This distinction resembles the notions of non-malleable commitments with respect to commitment and with respect to opening [14, 18]. Depending on the application scenario of non-malleable hash functions the stronger or weaker version might be required. As an example, the result about the Bellare-Rogaway encryption scheme uses the stronger definition above, and our construction in the next section achieves this stronger notion, which obviously implies the weaker one.

REMARK 4. Note that we only demand that $(x, y) \neq (x^*, y^*)$ for the adversary's choice (instead of demanding $x \neq x^*$ or $y \neq y^*$ instead), yielding a stronger definition, especially when the randomized hash function has multiple images for some input. Again, the particular need depends on the application and our solution meets this stronger requirement.

REMARK 5. In the case of non-malleable encryption the original simulation-based definition of [15] was later shown to be equivalent to an indistinguishability-based definition [5]. The superficial similarity between our definition of non-malleable hash functions and the one of non-malleable encryption suggests that this may be possible here as well. Surprisingly, straightforward attempts to define non-malleability of hash functions through indistinguishability do not seem to yield an equivalent definition. We discuss this issue in Appendix A , and leave it as an interesting open problem to find a suitable indistinguishability-based definition for non-malleable hash functions.

REMARK 6. The usual security notions for hash functions include one-wayness and collision-resistance. However, neither property is known to follow from Definition 3.1. Consider a *constant* function $\mathsf{H}$ which is clearly not one-way nor collision-resistant. But the function is weakly non-malleable as a simulator can simulate $\mathcal{A}$ in a black-box way by handing the adversary the constant value. We keep these rather orthogonal security properties separate, as some applications may require one but not the others.

REMARK 7. Some applications (like the HMAC example in [17]) require a multi-valued version of the definition in which the adversary can adaptively generate several distributions and receive the images (with side information) before deciding upon $y^*$. One can easily extend our definition accordingly, letting $\mathcal{A}_d$ loop several times, in each round $i$ generating a distribution $\mathcal{X}_i$ and receiving $y_i$ and $h_{x_i}$ at the beginning of the next round and before outputting an image $y^*$. In general, it is possible to extend our construction to this case using stronger, adaptive versions of POWHFs and NIZKPoKs. See Remark 1 after Theorem 4.2.

## 4 Constructing Non-Malleable Hash Functions

In this section we give feasibility results via constructions for non-malleable hash functions. The main ingredient of our constructions is a perfectly one-way hash function (POWHF) [8, 11], which hides all information about the pre-image but which may still be malleable [6]. To ensure non-malleability we tag the hash value with a simulation-sound non-interactive zero-knowledge proof of knowledge of the pre-image. We first recall the definitions of these two primitives.

For POWHFs we slightly adapt the definition from [8, 11] to our setting. Originally, POWHFs have been defined to have a specific input distribution $\mathcal{X}$ (like the uniform distribution in [11, 16]). Here we let the adversary choose the input distribution adaptively, and merely demand that this distribution $\mathcal{X}$ satisfies a certain efficient predicate $P_{\mathsf{pow}}(\mathcal{X})$; this is analogous to the non-malleability experiment in which the adversary chooses $\mathcal{X}$ and the relation $R$ takes $\mathcal{X}$ as additional input. Also note that we call the side information here $\mathsf{aux}$ (as opposed to $\mathsf{hint}$ for non-malleability) in order to distinguish between the two primitives. In fact, in our construction $\mathsf{aux}$ uses $\mathsf{hint}$ as a sub routine but generates additional output.

**Definition 4.1 (POWHF)** *A hash function $\mathcal{P} = (\mathsf{POWK}, \mathsf{POW}, \mathsf{POWVf})$ is called a perfectly one-way hash function (with respect to predicate $P_{\mathsf{pow}}$ and probabilistic function $\mathsf{aux}$) if it is collision resistant, and if for any PPTA $\mathcal{B} = (\mathcal{B}_d, \mathcal{B}_b)$, where $\mathcal{B}_b$ has binary output, the following random variables are computationally indistinguishable:*

$$\begin{array}{l|l}
K \xleftarrow{\$} \mathsf{POWK}(1^k)\ ;\ x \xleftarrow{\$} \mathcal{X}(1^k) & K \xleftarrow{\$} \mathsf{POWK}(1^k) \\
a_x \xleftarrow{\$} \mathsf{aux}(K,x)\ ;\ y \xleftarrow{\$} \mathsf{POW}(K,x) & (\mathcal{X}, st_d) \xleftarrow{\$} \mathcal{B}_d(K)\ ;\ x \xleftarrow{\$} \mathcal{X}(1^k),\ x' \xleftarrow{\$} \mathcal{X}(1^k) \\
b \xleftarrow{\$} \mathcal{B}_b(y, a_x, st_d) & a_x \xleftarrow{\$} \mathsf{aux}(K,x)\ ;\ y' \xleftarrow{\$} \mathsf{POW}(K,x') \\
\textit{return } (K,x,b) \textit{ if } P_{\mathsf{pow}}(\mathcal{X}) = 1, \textit{ else } \bot & b \xleftarrow{\$} \mathcal{B}_b(y', a_x, st_d) \\
 & \textit{return } (K,x,b) \textit{ if } P_{\mathsf{pow}}(\mathcal{X}) = 1, \textit{ else } \bot
\end{array}$$

REMARK 1. As pointed out in [8, 11] the definition only makes sense if $\mathsf{aux}$ is an uninvertible function of the input (such that finding the pre-image $x$ from $a_x$ is infeasible) and $\mathcal{B}_x$ only outputs descriptions of well-spread distributions (with super-logarithmic min-entropy). Otherwise the notion is impossible to achieve. For generality, we do not restrict $\mathcal{X}$ and $\mathsf{aux}$ explicitly here.

REMARK 2. Perfectly one-way hash functions (in the sense above) can be constructed from any one-way permutation [11, 16] (for the uniform input distribution), any regular collision-resistant hash function [11] (for any distribution with fixed, super-logarithmic min-entropy), or under the decisional Diffie-Hellman assumption [8] (for the uniform distribution). Usually these general constructions are not known to be secure assuming arbirtrary functions $\mathsf{aux}$, yet for the particular function $\mathsf{aux}$ required by the application they can often be adapted accordingly. A concrete example is given in Section 6, in our discussion of the Bellare-Rogaway encryption scheme.

ON THE CHOICE OF THE RELATION CLASS. Recall that the definition of non-malleability is parametrized by a class of relations. As explained earlier in the paper, no non-malleable hash function for an arbitrary class exists (see Remark 1 after Definition 3.1). In the sequel, we exhibit a class of relations for which we show how to construct non-malleable hash functions, and then present our provably secure construction.

Specifically, we consider the class of relations $\mathcal{R}_{\mathsf{pred}}^{\mathsf{rinfo}}$, parameterized by an optional function $\mathsf{rinfo}$ and which consists of all relations of the form $R(x, x^*) = P(x, P^*(\mathsf{rinfo}(x), x^*))$, for all efficient predicates $P, P^*$.[4] The function $\mathsf{rinfo}(x)$ may be empty or consist of a small fraction of bits of $x$ (e.g., up to logarithmically many), and should be interpreted as the information about $x$ that may be used in evaluating the relation $R$. It is important that $\mathsf{rinfo}$ is an univertible function, as otherwise, if one could recover $x$ from $\mathsf{rinfo}(x)$, then $\mathcal{R}_{\mathsf{pred}}^{\mathsf{rinfo}}$ would comprise all efficient relations, $R(x, x^*) = P^*(x, x^*)$, and non-malleability with respect to this class, again, would not be achievable.

As an example consider the empty function $\mathsf{rinfo}$ such that $\mathcal{R}_{\mathsf{pred}}$ consists of all relations $R(x, x^*) = P(x, P^*(x^*))$. This class of relations allows to check for instance that individual bits of $x$ and $x^*$ are complement of each other, i.e., if $\pi_j$ denotes the projection onto the $j$-th bit then one sets $P^*(x^*) = \pi_j(x^*)$ and lets

[4] Where we neglect the distribution $\mathcal{X}$ as part of the relation's input for the moment.

$P(x, P^*(x^*))$ output 1 if $\pi_j(x) \neq \pi_j(x^*)$. This example has also been used by Boldyreva and Fischlin [6] to show the necessity of non-malleability for OAEP, and to give an example of a perfectly one-way hash function that is malleable in the sense that flipping the first bit of an image produces a hash of the pre-image whose first bit is also flipped.

In the examples above rinfo has been the empty function. Of course, using non-trivial functions rinfo allows for additional relations and enriches the class $\mathcal{R}_{\mathsf{pred}}^{\mathsf{rinfo}}$. Consider for example a hash function $H$ that is malleable in the sense that an adversary, given $H(K, r\|m)$ for random $r \in \{0,1\}^k$, can compute $H(K, r\|m')$ for some $m' \neq m$. One way to capture that the two pre-images coincide on the first $k$ bits is to set $\mathsf{rinfo}(r\|m) = r$ and to set $P^*(r, x^*) = 1$ if and only if $r$ is the prefix of $x^*$. Since rinfo should be univertible, the function should rather return only a fraction of $r$, though. Similarly, one can see that the class $\mathcal{R}_{\mathsf{pred}}^{\mathsf{rinfo}}$ "captures" relations like $R(x, x^*) = 1$ iff $x \oplus x^* = \delta$ for some constant $\delta$, and many other useful relations.

Finally, we note that each relation from the class also checks that the chosen input distribution $\mathcal{X}$ "complies" with the eligible distributions from the underlying POWHF. That is, each relation also checks that the predicate $P_{\mathsf{pow}}(\mathcal{X})$ of the POWHF is satisfied. The full relation $R(\mathcal{X}, x, x^*)$ then evaluates to 1 iff $P(x, P^*(\mathsf{rinfo}(x), x^*)) = 1$ and $P_{\mathsf{pow}}(\mathcal{X}) = 1$. More formally, for any predicate $P_{\mathsf{pow}}$ and uninvertible function rinfo we define the class of relations:

$$\mathcal{R}_{\mathsf{pred}}^{\mathsf{rinfo}, P_{\mathsf{pow}}} = \left\{ R : \begin{array}{l} \text{there exist efficient (probabilistic) predicates } P, P^* \\ \text{such that } R(\mathcal{X}, x, x^*) = P(x, P^*(\mathsf{rinfo}(x), x^*)) \wedge P_{\mathsf{pow}}(\mathcal{X}) \end{array} \right\}.$$

Our construction also uses a simulation-sound zero-knowledge proof of knowledge $\Pi = (\mathsf{CRS}, \mathsf{P}, \mathsf{V})$ for the NP-relation $R_{\mathsf{pow}}$ defined by:

$$R_{\mathsf{pow}} = \{(K_{\mathsf{pow}}||y_{\mathsf{pow}}, x||r) : \mathsf{POW}(K_{\mathsf{pow}}, x; r) = y_{\mathsf{pow}}\}.$$

which essentially says that one "knows" a pre-image of a hash value. Simulation-sound NIZK proofs of knowledge for such relations can be derived from trapdoor permutations [27, 12]. We recall the definition of such proof systems in Appendix B.

THE CONSTRUCTION AND ITS SECURITY. The following theorem captures the security of our construction.

**Theorem 4.2** *Let $\mathcal{P} = (\mathsf{POWK}, \mathsf{POW}, \mathsf{POWVf})$ be a perfectly one-way hash function with respect to $P_{pow}$ and* aux*, where* aux $= (\mathsf{hint}, \mathsf{rinfo})$ *for probabilistic functions* hint *and* rinfo*. Let $\Pi = (\mathsf{CRS}, \mathsf{P}, \mathsf{V})$ be a simulation-sound non-interactive zero-knowledge proof of knowledge for relation $R_{pow}$. Then the following hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ is non-malleable with respect to* hint *and $\mathcal{R}_{pred}^{\mathsf{rinfo}, P_{\mathsf{pow}}}$:*

- *PPTA* HK *on input $1^k$ samples $K_{pow} \xleftarrow{\$} \mathsf{POWK}(1^k)$ and $\mathrm{crs} \xleftarrow{\$} \mathsf{CRS}(1^k)$ and outputs $K = (K_{pow}, \mathrm{crs})$. The associated domain $D_K$ is given by $D_{K_{pow}}$.*
- *PPTA* H *on input $K$ and $x \in D_K$ computes $y_{pow} \leftarrow \mathsf{POW}(K_{pow}, x; r)$ for random $r \xleftarrow{\$} RND_{K_{pow}}$ as well as $\pi \xleftarrow{\$} \mathsf{P}(\mathrm{crs}, K_{pow}||y_{pow}, x||r)$. It outputs $y = (y_{pow}, \pi)$.*
- *PTA* HVf *for inputs $K = (K_{pow}, \mathrm{crs})$, $x$ and $y = (y_{pow}, \pi)$ outputs 1 if and only if* $\mathsf{POWVf}(K_{pow}, x, y_{pow}) = 1$ *and* $\mathsf{V}(\mathrm{crs}, K_{pow}||y_{pow}, \pi) = 1$.

*In addition, $\mathcal{H}$ is collision-resistant.*

We give the proof of the theorem in Appendix C. Here we provide a sketch. Consider an adversary $\mathcal{A}$ attacking the non-malleability property, i.e., getting $(y, h_x, \mathrm{st}_d)$ as input (after picking distribution $\mathcal{X}$) and subsequently producing outputs $(y^*, \mathrm{st}_y)$ and $x^*$ (after learning $x$). Then we construct a simulator $\mathcal{S}$ with black-box access to $\mathcal{A}$ as follows. $\mathcal{S}$ first simulates $\mathcal{A}$ to produce distribution $\mathcal{X}$. In the second stage, $\mathcal{S}$ computes the POWHF value $y'_{\mathsf{pow}}$ of an independent sample $x'$ and then prepares a zero-knowledge proof $\pi'$ for this value. It runs $\mathcal{A}$ on this value $y' = (y'_{\mathsf{pow}}, \pi')$ to receive $y^*$ for $y^* = (y^*_{\mathsf{pow}}, \pi^*)$. The simulator then uses the knowledge extractor of the NIZKPoK to recover $x^*$ from $y^*_{\mathsf{pow}}, \pi^*$ and outputs this value $x^*$.

By the perfect one-wayness of the POWHF (with respect to $\mathsf{aux}$) and the zero-knowledge property of the proof, running $\mathcal{A}$ on the fake value $y'$ cannot change $\mathcal{A}$'s success probability significantly. By the simulation soundness of the NIZK the simulator is most likely able to extract the pre-image $x^*$ from $y^*$ (even if it has generated only a simulated proof $\pi'$ before). The collision-resistance of the POWHF finally implies that this value $x^*$ is quasi unique and will satisfy the non-malleability relation $R$ with essentially the same probability.

REMARK 1. The malleability adversary has access to essentially two different sources of partial information about $x$: $\mathsf{hint}(x)$ which it receives explicitly as input, and $\mathsf{rinfo}(x)$ which it can use indirectly through the relation $R$. This motivates the requirement that $\mathcal{P}$ be perfectly one-way with respect to partial information $\mathsf{aux} = (\mathsf{hint}, \mathsf{rinfo})$.

REMARK 2. As mentioned after the definition of non-malleable hash functions, some applications (like the one about HMAC [17]) may require a stronger notion in which the adversary can adaptively generate distributions and receives the images, before deciding upon $y^*$. Our construction above can be extended to this case, assuming that the POWHF obeys a corresponding "adaptiveness" property and that the zero-knowledge proof of knowledge is multiple simulation-sound and multiple zero-knowledge. Such adaptively-secure POWHFs (for uniform distributions) can be built from one-way permutations [16] and suitable zero-knowledge proofs exist, assuming trapdoor permutations [27, 12].

# 5 On the Complexity of Non-Malleable Functions

In this section we discuss the existential complexity of non-malleable functions. We first indicate, via an oracle separation result, that deriving non-malleable hash and one-way functions via one-way permutations is infeasible. We then continue to show that non-malleable hash functions imply POWHFs.

## 5.1 On the Impossibility of Black-Box Reductions

We first show that, under reasonable conditions, there is no black-box reduction from non-malleable hash functions (which might not even be collision-resistant but rather one-way only) to one-way permutations. For space reasons most of the proofs have been delegated to Appendix D.

BLACK-BOX REDUCTIONS. In their seminal paper Impagliazzo and Rudich [22] have shown that some cryptographic primitives cannot be derived from other primitives, at least if the starting primitive is treated as a black box. Instead of separating primitives as in [22] here we follow the more accessible approach of Hsiao and Reyzin [21], giving a relaxed separation result with respect to black-box security reductions. We give a formalization of the oracle-based black-box separation approach that we use in Appendix D.

For our result we assume that the algorithms of the hash function $\mathcal{H}$ are granted oracle access to a random permutation oracle $\mathcal{P}$ (which is one-way, of course). A black-box reduction to $\mathcal{P}$ is now an algorithm which, with oracle access to $\mathcal{P}$ and a putative successful attacker $\mathcal{A}$ on the non-malleability property, inverts $\mathcal{P}$ with noticeable probability. Such an attacker $\mathcal{A}$ may take advantage of another oracle $\mathcal{O}$ (related to $\mathcal{P}$) which allows it to break the non-malleability but does not help to invert the one-way permutation $\mathcal{P}$. Since neither the construction nor the reduction are given access to $\mathcal{O}$, the reduction must be genuinely black-box.

DEFINING ORACLES $\mathcal{P}$ AND $\mathcal{O}$. For now we let $\mathcal{P}$ be a random permutation oracle which in particular is a one-way function. Below we show through de-randomization techniques that some fixed $\mathcal{P}$ must also work. For our separation we let the side information of the non-malleable hash function include an image of the uniformly distributed input $x$ under $\mathcal{P}$. More precisely, consider the function $\mathsf{hint}^{\mathcal{P}}_{\mathsf{sep}}$ which on input $(1^k, K, x)$ for random $x$ computes $h_x = \mathcal{P}(0^k || x || \langle \mathsf{HVf} \rangle || K)$ for the description $\langle \mathsf{HVf} \rangle$ of the verification algorithm and finally outputs $h_x$.[5]

[5] We note that the side information $h_x$ does not reveal any essential information about $x$ in the sense that one can show that, for any

We next construct the oracle $\mathcal{O}$ that helps to break non-malleability. The idea is that using $\mathcal{O}$ it is possible to extract from the image $y$ and "hint" $h_x$ (described above) the pre-image $x$ of $y$. Since the adversary gets $y$ as input, but the simulator does not, the oracle is only helpful to the adversary. Note that breaking non-malleability means that no simulator of comparable complexity is able to approximate the success probability of $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ closely. To ensure that the simulator has the equal power as $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ we grant the simulator $\mathcal{S}^{\mathcal{P},\mathcal{O}}$ therefore access to both oracles $\mathcal{P}, \mathcal{O}$.

**Construction 5.1** *Let oracle $\mathcal{O}$ take as input a parameter $1^k$, an image $y$ and a "hint" $h_x$. The oracle first finds the pre-image $z||x||\langle \mathsf{HVf} \rangle ||K$ of $h_x$ under $\mathcal{P}$ and verifies that $z = 0^k$; if not it immediately returns $\perp$. Else it checks that $\mathsf{HVf}^{\mathcal{P}}(K, x, y) = 1$ and returns $x$ if so (and outputs $\perp$ otherwise).*

We show that $\mathcal{O}$ does not help to invert $\mathcal{P}$, thus showing that relative to the oracles there still exists one-way permutations (see again Appendix D for the proof):

**Proposition 5.2** *For any efficient algorithm $\mathcal{B}^{?,?}$, the probability that $\mathcal{B}^{\mathcal{P},\mathcal{O}}$ breaks the one-wayness of $\mathcal{P}$ is negligible.*

In light of this lemma we conclude that there exists a particular $\mathcal{P}$ that is hard to invert for all PPT adversaries with oracles $\mathcal{P}, \mathcal{O}$. The argument is the same as in [21]. For a fixed PPT adversary $\mathcal{B}$, we define the sequence of events (indexed by $k$) where $\mathcal{B}$ inverts strings of length $k$ with some good probability; for a suitable choice of parameters, the sum of the probabilities (over $\mathcal{P}$) of these events converges and by the first Borel-Cantelli lemma only finitely many of these events may occur, almost surely. Then taking the countable intersection over all PPT $\mathcal{B}$, we get that there is at least one $\mathcal{P}$ with the desired property.

SEPARATION. We require some mild, technical conditions for our non-malleable hash function and the relation. Namely, we assume that

- the hash function is *non-trivial* meaning that it is infeasible to predict an image for uniformly distributed input over $\{0,1\}^k$ (thus ruling out trivial examples like constant hash functions), and
- the relation class $\mathcal{R}$ contains the relation $R_{\mathsf{sep}}$ which on input $(\mathcal{X}, x, x^*)$ checks that $\mathcal{X}$ is the uniform distribution on $\{0,1\}^k$, and that $\mathsf{parity}(x) = \bigoplus x_i = \mathsf{parity}(x^*) = \bigoplus x_i^*$. Note that $R_{\mathsf{sep}} \in \mathcal{R}_{\text{pred}}$ for our predicate-based relations, even for the empty function $\mathsf{rinfo}$, and can thus be achieved in principle.

**Theorem 5.3** *Let $\mathcal{H}^{\mathcal{P}} = (\mathsf{HK}^{\mathcal{P}}, \mathsf{H}^{\mathcal{P}}, \mathsf{HVf}^{\mathcal{P}})$ be a non-trivial non-malleable hash function with respect to $\mathsf{hint}^{\mathcal{P}}_{\mathsf{sep}}$ and $\mathcal{R} \ni R_{\mathsf{sep}}$. Then there exists an adversary $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ that breaks non-malleability of $\mathcal{H}^{\mathcal{P}}$ (for any simulator $\mathcal{S}^{\mathcal{P},\mathcal{O}}$).*

The idea is that the adversary can easily compute the pre-image $x$ with the help of $y, h_x$ and oracle $\mathcal{O}$ and then find another value $x^*$ with the same parity as $x$. In constrast, the simulator most likely cannot get a useful answer from $\mathcal{O}$ given $h_x$ only (but not $y$), because it is infeasible to guess the right value $y$ by the non-triviality of the hash function. Hence, the adversary succeeds with probability $1$ while any simulator cannot be more successful than with probability close to $1/2$.

**Corollary 5.4** *There exists no black-box reduction from non-trivial non-malleable functions (with respect to $\mathsf{hint}^{\mathcal{P}}_{\mathsf{sep}}$ and $\mathcal{R} \ni R_{\mathsf{sep}}$) to one-way permutations.*

At first glance it seems as if our result would transfer (after some minor modifications) to other non-malleable primitives like commitments. This is not the case. The oracle $\mathcal{O}$ in our construction relies on the ability to check whether a pre-image $x$ matches an image $y$ (public verifiability of hash functions), while other primitives such as encryption $\mathcal{E}(m; r)$ and commitments $\mathrm{Com}(m; r)$ use hidden randomness (which is not part of the input of function $\mathsf{hint}$).

non-malleable hash function for the uniform input distribution and no side information at all, the hash function remains non-malleable with respect to $h_x$ relative to the random permutation $\mathcal{P}$ (but not relative to $\mathcal{O}$, of course). Also observe that the common strategy of using black-box simulators usually works for any side information, and in particular for the one here.

### 5.2 On the Relation between Non-Malleability and Perfect One-Wayness

It it intuitively appealing to conclude that a function which is not perfect one-way is also malleable. Roughly, if an adversary can recover even a single bit of information about $x$, say $P(x)$, from the hash value $\mathsf{H}(x)$, then it should be able to produce $\mathsf{H}(x^*)$ for some $x^*$ such that $P(x^*) = P(x)$. As long as the distribution from which $x$ is selected has enough min-entropy, it would be more difficult for a simulator, not seeing $\mathsf{H}(x)$, to emulate the behavior of the adversary. The following proposition captures the above intuition, using an alternative formulation for perfect one-wayness, called oracle-simulatability [8]. When no auxiliary information is present, perfect one-wayness in the oracle simulatability sense is equivalent to perfect one-wayness in the sense of Definition 4.1 for *non-uniform* adversaries [8]. In general however, security in Definition 4.1 is only known to imply security in the oracle simulatability sense, and not vice versa. In Appendix E we recall the oracle-simulatability based definition of perfect one-wayness and prove the following:

**Proposition 5.5** *Let $\mathcal{H}$ be a hash function that is collision resistant and non-malleable with respect to side information* $\mathsf{hint}$ *and relation class*[6] $\mathcal{R}_{pred}^{P_{\mathsf{pow}}}$*, where $P_{\mathsf{pow}}$ is such that $P_{\mathsf{pow}}(\mathcal{X}) = 1$ implies that $\mathcal{X}$ is well-spread. Then $\mathcal{H}$ is perfectly one-way with respect to $P_{\mathsf{pow}}$ and partial information* $\mathsf{hint}$*, in the oracle-simulatability sense.*

## 6 Applications

In this section we study the usefulness of our notion for cryptographic applications. As an example we show that when one of the two random oracles in the aforementioned encryption scheme proposed by Bellare and Rogaway in [4] is instantiated with a non-malleable hash function, the scheme remains IND-CCA secure. In addition, we argue that non-malleability is useful in preventing off-line computation attacks against a certain class of cryptographic puzzles.

INSTANTIATING RANDOM ORACLES. We start with recalling the scheme. Let $\mathcal{F}$ be a familiy of trapdoor permutations and $G, H$ be random oracles. The message space of the scheme $\mathrm{BR}^{G,H}[\mathcal{F}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is the range of $G$. The key generation algorithm $\mathcal{K}$ outputs a random $\mathcal{F}$-instance $\mathsf{f}$ and its inverse $\mathsf{f}^{-1}$ as the public and secret key, respectively. The encryption algorithm $\mathcal{E}$ on inputs $\mathsf{f}$ and $m$ picks random $r$ in the domain of $\mathsf{f}$ (we assume that $r \in \{0,1\}^k$) and outputs $(\mathsf{f}(r), G(r) \oplus m, H(r\|m))$. The decryption algorithm on inputs $\mathsf{f}^{-1}$ and $(y, g, h)$ first computes $r \leftarrow \mathsf{f}^{-1}(y)$, then $m \leftarrow g \oplus G(r)$, and outputs $m$ iff $H(r\|m) = h$. The scheme $\mathrm{BR}^{G,H}[\mathcal{F}]$ is proven to be IND-CCA secure in the random oracle model assuming that $\mathcal{F}$ is one-way [4].

Here we study the possibility of realizing the random oracle $\mathcal{H}$ with an actual hash function family $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$, a so-called *partial $H$-instantiation* of the scheme. More precisely, we modify the scheme so that the public key and secret key also contain a key $K \xleftarrow{\$} \mathsf{HK}(1^k)$ specifying a function. Then $\mathcal{E}$ computes $\mathsf{H}(K, r\|m)$ instead of $H(r\|m)$, and $\mathcal{D}$ computes $\mathsf{HVf}(K, r\|m, h)$ instead of checking that $H(r\|m) = h$. We refer to this scheme as $\mathrm{BR}^{G,\mathcal{H}}[\mathcal{F}]$. As explained in the Introduction, non-malleability is a necessary property for $\mathcal{H}$ for the scheme to be IND-CCA secure (still in the random oracle model). The following shows that non-malleability is in fact sufficient for a secure partial $H$-instantiation.

Before stating the sufficient conditions for security to hold, we fix some notation. Below we let the function $\mathsf{rinfo}_{\mathsf{BR}}(x) = \mathrm{msb}_{k/2}(x)$ output the $k/2$ most significant bits of its input. The class of relations we require here for non-malleability is only a subset of the achievable class discussed in Section 4. Namely, we only require a relation of the form $R_{\mathsf{BR}}(\mathcal{X}, x, x^*) = P^*(\mathsf{rinfo}_{\mathsf{BR}}(x), x^*) \wedge P_{\mathsf{pow}}(\mathcal{X})$, where $P_{\mathsf{pow}}$ is the predicate that checks that $\mathcal{X}$ is the canonical representation of the uniform distribution on the first $k$ bits, and $P^*$ is the predicate that simply verifies that $\mathrm{msb}_{k/2}(x^*) = \mathsf{rinfo}_{\mathsf{BR}}(x)$. We choose this specific predicate $R_{\mathsf{BR}}$ so that it can check if $x = x^*$, while erring with only negligible probability, but still admit the construction of non-malleable hash functions.

[6] With empty function $\mathsf{rinfo}$.

Below we will require that the trapdoor permutation family is $\mathrm{msb}_{k/2}$-*partial one-way*, meaning that it is hard to compute the $k/2$ most significant bits of the random input $r$ given a random instance $\mathsf{f}$ and $\mathsf{f}(r)$ (cf. [19] for the formal definition). This is a rather mild assumption to impose on $\mathcal{F}$. For example, RSA was shown to be partial one-way under the RSA assumption in [19]. A general approach to construct such a partial one-way family $\mathcal{F}$ is to define $\mathsf{f}(r) = g(\mathrm{msb}_{k/2}(r))\|g(\mathsf{lsb}_{k/2}(r))$ for a trapdoor permutation $g$.[7]

We need one more technical detail before stating the theorem. We start with some hash function family $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ and trapdoor permutation family $\mathcal{F}$. We will require that $\mathcal{H}$ is non-malleable, even when a random instance of $\mathcal{F}$ is included with the key output by $\mathsf{HK}$. We stress that this detail still leaves our non-malleable hash function achievable by the construction in the previous section. We write $\mathcal{H} = (\mathsf{HK}_{\mathcal{F}}, \mathsf{H}, \mathsf{HVf})$ for the modified hash function for which key generation outputs a random instance of $\mathcal{F}$ along with the original hash key. The purpose of this change is to allow the side information function to compute a hint related to the random trapdoor permutation instance. Below we write $\mathsf{hint}_{\mathsf{BR}}$ for the function that takes as input a key $(K, \mathsf{f})$ and string $x$, and outputs $\mathsf{f}(r)$, where $r$ are the first $k$ bits of the input $x$. We note the IND-*CPA* version of the scheme by Bellare and Rogaway was shown secure in the standard model by Canetti [8], assuming the hash function is a POWHF with respect to a similar hint function.

**Theorem 6.1** *Let $\mathcal{F}$ be an $\mathrm{msb}_{k/2}$-partial one-way trapdoor permutation family and let $\mathcal{H} = (\mathsf{HK}_{\mathcal{F}}, \mathsf{H}, \mathsf{HVf})$ be a collision-resistant hash function which is non-malleable with respect to the function $\mathsf{hint}_{BR}$ and to the relation $R_{BR}$. Assume further that $\mathcal{H}$ is a perfectly one-way hash function with respect to $P_{pow}$ and $\mathsf{hint}_{BR}$. Then BR$^{G,\mathcal{H}}[\mathcal{F}]$ is IND-CCA secure (in the RO model).*

REMARK. Although the non-malleability property of the hash implies that no partial information about pre-images is leaked (cf. Theorem 5.5 for a formal statement of this implication), the theorem above requires the hash to be perfectly one-way in the sense of Definition 4.1, which is a stronger requirement in general.

The proof is in Appendix F. Here we provide some intuition for it. Consider an adversary $\mathcal{B}$ that breaks IND-CCA security of the scheme. After selecting two messages $m_0, m_1$ it is given the challenge ciphertext of the form $(y, g, h) = (\mathsf{f}(r), G(r) \oplus m_b, \mathsf{H}(K, r\|m_b))$ for a random string $r$ and bit $b$, and $\mathcal{B}$ tries to predict $b$. We first claim that the scheme is IND-CPA, meaning that without decryption queries $\mathcal{B}$ cannot break security. This follows from the perfect one-wayness condition above. That is, if $\mathcal{B}$ has non-negligible advantage in determining $b$ without making any decryption queries, then one can break perfect one wayness of $\mathcal{H}$.

Next we show that decryption queries are useless to $\mathcal{B}$. Assume that $\mathcal{B}$ makes decryption queries of the form $(y', g', h')$. If $\mathcal{B}$ has queried oracle $G$ about $r' = \mathsf{f}^{-1}(y')$ before then we can easily find this entry in the list of $G$-queries and simulate the additional decryption steps. Else, consider the case that $\mathcal{B}$ has not made such a query to $G$ but tries to succeed by mauling the challenge ciphertext $(y, g, h)$ to $(y, g', h')$. Then it follows from the non-malleability of $\mathcal{H}$ that this ciphertext is likely to be invalid. The collision-resistance of $\mathcal{H}$ additionally prevents the case that $\mathcal{B}$ creates any other valid ciphertext $(y', g', h)$ without querying $G$ about $r'$ before.

APPLICATION TO CRYPTOGRAPHIC PUZZLES. Cryptographic puzzles are a defense mechanism against denial of service attacks (DoS). The idea is that, before spending any resources for the execution of a session between a client and a server, the server requires the client to solve a puzzle. Since solving puzzles requires spending cycles, the use of puzzles prevents a malicious client to engage in a large number of sessions without spending itself a significant amount of resources. One desirable condition is that the server does not store any client-related state.

A simple construction for such puzzles proposed by Juels and Brainard [23] is based on any arbitrary one-way function $h : \{0,1\}^l \to \{0,1\}^l$. First, select at random $x \xleftarrow{\$} \{0,1\}^l$ and compute $y = h(x)$. Then, a puzzle is given by the tuple $(x[1..l-k], y)$ consisting of the first $l-k$ bits of $x$ together with $y$. To prove it solved the

[7] In fact, this construction also has the useful property that $\mathsf{f}(r)$ is still hard to invert, even if given $\mathrm{msb}_{k/2}(r)$. Thus this trapdoor permutation is suitable for constructing POWHFs secure with respect to side information $(\mathrm{msb}_{k/2}(r), \mathsf{f}(r))$ and therefore, via our construction, non-malleable hash functions for side information $\mathsf{hint}_{\mathsf{BR}}(r) = \mathsf{f}(r)$ and the relation $R_{\mathsf{BR}}$. In other words, non-malleable hash functions for $\mathsf{hint}_{\mathsf{BR}}$ and $R_{\mathsf{BR}}$ exist under common cryptographic assumptions.

puzzle, the client has to return $(x, y)$. It can be easily seen that the construction above is not entirely satisfactory. In particular, it either fails against replay attacks —where the clients present the same puzzle-solution pair to the server— or the server needs to store all of the $x$'s used to compute the puzzles.

The solution proposed to mitigate the above problem is to compute $x$ as $H(S, t)$, where $S$ is some large bitstring known only to the server, and $t$ is some bitstring that somehow "expires" after a certain amount of time (this can be for example the current system time). The puzzle is then given by $(t, x[1..l - k], y)$, where $y = h(x)$. A solution (or solved puzzle) is $(t, x, y)$ which needs to satisfy the obvious equations, and moreover, $t$ is not an expired bitstring.

In the setting above, non-malleability of $H$ surfaces as an important property. If out of the first two elements $(t, H(S, t))$ of a puzzle solution the adversary can efficiently construct $(t', H(S, t'))$ for $t' \neq t$, a string which has not yet expired, then the defense sketched above is rendered useless: the adversary can easily construct new puzzles (together with their solutions). Requiring that the function $H$ is non-malleable with respect to the relation $R(s_1, s_2) = 1$ iff $s_1 = (S, t)$ and $s_2 = (S, t')$ for $t \neq t'$ is sufficient to prevent the above attack.

## 7 Conclusions

We initiated the study of non-malleability of hash and one-way functions. We designed a definition of non-malleability and showed that it can be met. Namely, we proposed a (theoretical) construction from perfect one-way hash functions and simulation-sound non-interactive zero-knowledge proofs of knowledge. We discussed the complexity of non-malleable functions, and gave a black-box based separation of non-malleable functions from one-way permutations. We exemplified the usefulness of our definition in cryptographic applications by showing that non-malleability is necessary and sufficient to securely replace one of the two random oracles in the IND-CCA encryption scheme by Bellare and Rogaway, and to improve the security of client-server puzzles. We believe that our definition will find other interesting applications, and while our treatment is mostly theoretical, it helps to understand a practical property that designers of hash functions can keep in mind.

## Acknowledgments

We thank Vipul Goyal for discussions.

Alexandra Boldyreva and David Cash are supported in part by Alexandra's NSF CAREER award 0545659 and NSF Cyber Trust award 0831184. Marc Fischlin is supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

## References

[1] B. Barak. Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. FOCS, IEEE, 2002.

[2] B. Barak and M. Prabhakaran and A. Sahai. Concurrent Non-Malleable Zero Knowledge. (FOCS), pp. 563–572, IEEE, 2005.

[3] M. Bellare and R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. Crypto'96, Vol. 1109 of LNCS, pp. 1–15, Springer-Verlag, 1996.

[4] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, *CCS*, ACM, 1993.

[5] M. Bellare and A. Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. Crypto'99, Vol. 1666 of LNCS, pp. 519–536, Springer-Verlag, 1999.

[6] A. Boldyreva and M. Fischlin. Analysis of random-oracle instantiation scenarios for OAEP and other practical schemes. Crypto '05, Vol. 3621 of LNCS, pp. 412–429, Springer-Verlag, 2005.

[7] A. Boldyreva and M. Fischlin. On the Security of OAEP. Asiacrypt '06 Proceedings, Vol. 4284 of LNCS, Springer-Verlag, 2005.

[8] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. Crypto'97, Vol. 1294 of LNCS, pp. 455–469, Springer-Verlag, 1997.

[9] R. Canetti and R. R. Dakdouk. Extractable Perfectly One-Way Functions. ICALP'08, Vol. Volume 5126 of LNCS, pp. 449–460, Springer-Verlag, 2008.

[10] R. Canetti, S. Halevi and M. Steiner. Mitigating Dictionary Attacks on Password-Protected Local Storage. CRYPTO '06, Vol. 4117 of LNCS, pp. 160-179. Springer-Verlag, 2006.

[11] R. Canetti, D. Micciancio and O. Reingold. Perfectly one-way probabilistic hash functions. In *STOC '98*, pp. 131–140, ACM, 1998.

[12] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. *Robust Non-interactive Zero Knowledge*. Crypto 2001, Volume 2139 of LNCS, pages 566–598. Springer-Verlag, 2001.

[13] I. Damgård and J. Groth. Non-interactive and Reusable Non-Malleable Commitment Schemes. STOC, pp. 426–437, ACM, 2003.

[14] G. Di Crescenzo and Y. Ishai and R. Ostrovsky. Non-interactive and Non-Malleable Commitment. STOC, pp. 141–150, ACM Press, 1998.

[15] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. SIAM Journal on Computing, Volume 30(2), pp. 391–437, 2000.

[16] M. Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. Eurocrypt'99, Vol. 1592 of LNCS, pp. 429–444, Springer-Verlag, 1999.

[17] M. Fischlin. Security of NMAC and HMAC based on Non-Malleability. RSA-CT'08, LNCS, Springer-Verlag, 2008.

[18] M. Fischlin and R. Fischlin. Efficient Non-Malleable Commitment Schemes. Crypto 2000, Vol. 1880 of LNCS, pp. 414–432, Springer-Verlag, 2000.

[19] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern RSA-OAEP Is Secure under the RSA Assumption. Crypto '01, Vol. 2139 of LNCS, pp. 260–274, Springer-Verlag, 2001.

[20] O.Goldreich. The Foundations of Cryptography. (Volume 1), Cambridge University Press, 2004.

[21] C.-Y. Hsiao and L. Reyzin. Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins. Crypto '04, Vol. 3152 of LNCS, pp. 92-105, Springer-Verlag, 2004.

[22] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *STOC'89*, pp. 44–61, ACM, 1989.

[23] A. Juels and J. Brainard. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks In *NDSS'99*, pp. 151-165, 1999.

[24] O. Pandey, R. Pass and V. Vaikuntanathan. Adaptive One-Way Functions and Applications. Crypto'08, Vol. 5157 of LNCS, pp. 57–74, Springer-Verlag, 2008.

[25] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. FOCS, pp. 563–572, IEEE, 2005.

[26] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. STOC '05, ACM Press, 2005.

[27] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. FOCS '99. IEEE, 1999.

# A Towards Alternative Definitions of Hash Non-Malleability

In this section we describe some approaches to defining non-malleability of hash functions in the style of indistinguishability and show why these approaches fail. First consider the following indistinguishability definition:

**Definition A.1** Let $\mathcal{H} = (\mathsf{HK}, \mathcal{H}, \mathsf{HVf})$ be a hash function. It is *non-malleable with respect to side information* $\mathsf{hint}$ *and class of relations* $\mathcal{R}$ if for any adversary $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$ and relation $R \in \mathcal{R}$, $\mathbf{Adv}^{\mathrm{cnmh}}_{\mathcal{H},\mathcal{A},R}(k)$ is negligible, where

$$\mathbf{Adv}^{\mathrm{cnmh}}_{\mathcal{H},\mathcal{A},R}(k) = \Pr\left[\, \mathbf{Exp}^{\mathrm{cnmh}-1}_{\mathcal{H},\mathcal{A},R}(k) = 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\mathrm{cnmh}-0}_{\mathcal{H},\mathcal{A},R}(k) = 1 \,\right]$$

and

```
*Experiment* $\mathbf{Exp}^{\mathrm{cnmh}-1}_{\mathcal{H},\mathcal{A},R}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{A}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k)\ ;\ h_x \leftarrow \mathsf{hint}(x)$
    $y \leftarrow \mathcal{H}(K, x)$
    $(y^*, st_y) \leftarrow \mathcal{A}_y(y, h_x, st_d)$
    $x^* \leftarrow \mathcal{A}_x(x, st_y)$
    return *1 iff*
        $(y^* \neq y) \wedge \mathsf{HVf}(K, x^*, y^*) = 1$
        $\wedge R(\mathcal{X}, x, x^*) = 1$
```

```
*Experiment* $\mathbf{Exp}^{\mathrm{cnmh}-0}_{\mathcal{H},\mathcal{A},R}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{A}_d(K)$
    $x, \widehat{x} \xleftarrow{\$} \mathcal{X}(1^k)\ ;\ h_x \leftarrow \mathsf{hint}(x)$
    $y \leftarrow \mathcal{H}(K, x)$
    $(y^*, st_y) \leftarrow \mathcal{A}_y(y, h_x, st_d)$
    $x^* \leftarrow \mathcal{A}_x(x, st_y)$
    return *1 iff*
        $(y^* \neq y) \wedge \mathsf{HVf}(K, x^*, y^*) = 1$
        $\wedge R(\mathcal{X}, \widehat{x}, x^*) = 1$
```

Intuitively, this definition states that whatever value $\mathcal{A}$ can produce a hash of with the help of $y$, the resulting pre-image will be unrelated to the challenge message. Similarly to Definition 3.1, this definition is impossible to achieve for an arbitrary $R$: an adversary can always set $x^* = (K, y)$ and the relation $R(x, x^*) = R(x, (K, y)) = \mathsf{HVf}(K, x, y)$ will give $A$ non-negligible advantage for any $\mathcal{H}$. In this case $\mathbf{Exp}^{\mathrm{cnmh}-1}_{\mathcal{H},\mathcal{A},R}(k) = 1$ always, but $\Pr\left[\, \mathbf{Exp}^{\mathrm{cnmh}-0}_{\mathcal{H},\mathcal{A},R}(k) = 1 \,\right]$ is likely to be negligible.

To prevent the definition from being trivially unsatisfiable, we can try to limit $\mathcal{R}$ so that the relations only look at parts of $x^*$, as we do in Theorem 4.2. But $\mathcal{R}$ may still get some information about $x$ from an adversary that uses $h_x$ to compute $x^*$, and we must rule out the possibility that $R$ can simply tell whether such side information matches $x$ or not. Thus this definition only makes sense when we require that $\mathcal{R}$ cannot distinguish $h_x$ from the side information corresponding to another random message. This assumption is too strong for our purposes, such as in our encryption example. (We note that this does cause a problem in the simulation-based definition of non-malleability because such an attack is just using the side information, and a simulator can also do this.)

It is possible to define non-malleable hash security in a way that does not rule out maulings that depend on the actual hash value itself. This weaker version of security may be sufficient for some applications.

**Definition A.2** *Let* $\mathcal{H} = (\mathsf{HK}, \mathcal{H}, \mathsf{HVf})$ *be a hash function. It is* weakly non-malleable *if for any adversary* $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$*,* $\mathbf{Adv}^{\mathrm{cnmh}}_{\mathcal{H},\mathcal{A}}(k)$ *is negligible, where*

$$\mathbf{Adv}^{\mathrm{cnmh}}_{\mathcal{H},\mathcal{A}}(k) = \Pr\left[\, \mathbf{Exp}^{\mathrm{cnmh}-1}_{\mathcal{H},\mathcal{A}}(k) = 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\mathrm{cnmh}-0}_{\mathcal{H},\mathcal{A}}(k) = 1 \,\right]$$

*and*

```
**Experiment** $\mathbf{Exp}^{\mathrm{cnmh}-0}_{\mathcal{H},\mathcal{A}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{A}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k)$ ; $y \leftarrow \mathcal{H}(K, x)$
    $y^* \leftarrow \mathcal{A}_y(y, st_d)$
    $x^* \leftarrow \mathcal{A}_x(x)$
    return *1 iff*
        $(y^* \neq y) \wedge \mathsf{HVf}(K, x^*, y^*) = 1$
```

```
**Experiment** $\mathbf{Exp}^{\mathrm{cnmh}-1}_{\mathcal{H},\mathcal{A}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{A}_d(K)$
    $x, \widehat{x} \xleftarrow{\$} \mathcal{X}(1^k)$ ; $y \leftarrow \mathcal{H}(K, x)$
    $y^* \leftarrow \mathcal{A}_y(y, st_d)$
    $x^* \leftarrow \mathcal{A}_x(\widehat{x})$
    return *1 iff*
        $(y^* \neq y) \wedge \mathsf{HVf}(K, x^*, y^*) = 1$
```

Note that $\mathcal{A}_x$ *only* gets $x$ or $\widehat{x}$ as input when computing $x^*$. In particular, $\mathcal{A}_x$ does not know the hash key and does not get any state information from the other stages of the adversary. Intuitively, the adversary is committed to some mauling strategy when it specifies $\mathcal{A}_x$, and then $\mathcal{A}_d$ and $\mathcal{A}_y$ will try to suceed with respect to that strategy.

To see that this definition is strictly weaker than Definition 3.1, consider the following function. Let $\mathcal{H}$ be a perfectly one-way hash function, and define $\mathcal{H}'(K, x) = \mathcal{H}(K, x) \| \mathcal{H}(K, x \oplus w) \| w$, where $w$ is a uniformly chosen bit string that is not all zeros. Given $\mathcal{H}'(K, x) = y \| y' \| w$, $\mathcal{A}_y$ can compute $\mathcal{H}'(K, x \oplus w)$ by simply outputting $y' \| y \| w$ and having $\mathcal{A}_x(x)$ output $x \oplus w$. But such an adversary does seem to be ruled out by the above definition, as $\mathcal{A}_x$ does not have any information about $w$.

Hence, we leave at as an open question to find a suitable indistinguishability-based definition for non-malleable hash functions.

## B Simulation-Sound Non-Interactive Zero-Knowledge Proofs of Knowledge

Here we give the definition of simulation-sound NIZK proofs of knowledge [27, 12].

**Definition B.1 (SS-NIZK)** *A simulation-sound non-interactive zero-knowledge proof of knowledge* $\Pi = (\mathsf{CRS}, \mathsf{P}, \mathsf{V})$ *for* $\mathrm{NP}$*-relation* $R_L$ *consists of three PPTA, the common reference string generator* $\mathsf{CRS}$*, the prover* $\mathsf{P}$ *and the verifier* $\mathsf{V}$*, such that there exist a (pair of) PPTA* $\mathsf{Z} = (\mathsf{Z}_0, \mathsf{Z}_1)$*, the zero-knowledge simulator, and* $\mathsf{K}$*, the knowledge extractor, with the following properties:*

- *Completeness: For any security parameter* $k \in \mathbb{N}$*, any* $\mathrm{crs} \xleftarrow{\$} \mathsf{CRS}(1^k)$*, any* $(x, w) \in R_L$ *any* $\pi \xleftarrow{\$} \mathsf{P}(\mathrm{crs}, x, w)$ *we have* $\mathsf{V}(\mathrm{crs}, x, \pi) = 1$.
- *Zero-Knowledge: For any (pair of) PPTA* $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ *the following random variables are computationally indistinguishable:*

```
$\mathrm{crs} \xleftarrow{\$} \mathsf{CRS}(1^k)$
$(x, w, \mathit{state}) \xleftarrow{\$} \mathcal{D}_0(\mathrm{crs})$
*if* $(x, w) \in R_L$ *then* $\pi \xleftarrow{\$} \mathsf{P}(\mathrm{crs}, x, w)$
    *else* $\pi = \perp$
*return* $d \xleftarrow{\$} \mathcal{D}_1(x, w, \mathit{state}, \pi)$
```

```
$(\sigma, \mathrm{crs}) \xleftarrow{\$} \mathsf{Z}_0(1^k)$
$(x, w, \mathit{state}) \xleftarrow{\$} \mathcal{D}_0(\mathrm{crs})$
*if* $(x, w) \in R_L$ then $\pi \xleftarrow{\$} \mathsf{Z}_1(\sigma, x)$
    *else* $\pi = \perp$
*return* $d \xleftarrow{\$} \mathcal{D}_1(x, w, \mathit{state}, \pi)$
```

- *Simulation-Soundness: For any (pair of) PPTA* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *the following experiment returns* $1$ *with negligible probability only:*

```
$(\sigma, \mathrm{crs}) \xleftarrow{\$} \mathsf{Z}_0(1^k)$
$(x, \mathit{state}) \xleftarrow{\$} \mathcal{A}_0(\mathrm{crs})$
$\pi \xleftarrow{\$} \mathsf{Z}_1(\sigma, x)$
$(x^*, \pi^*) \xleftarrow{\$} \mathcal{A}_1(\mathit{state}, \pi)$
$w^* \xleftarrow{\$} \mathsf{K}(\sigma, x^*, \pi^*)$
*return* $1$ *iff* $(x^*, \pi^*) \neq (x, \pi)$ and $\mathsf{V}(\mathrm{crs}, x^*, \pi^*) = 1$ *and* $(x^*, w^*) \notin R_L$.
```

The proof is called *multiple* zero-knowledge, if one cannot even distinguish $\mathcal{D}$'s output in the case when $\mathcal{D}$ asks the prover $\mathsf{P}$ to see several proofs for adaptively chosen statements $(x, w)$, and in the case when the proofs are provided by the simulator $\mathsf{Z}_1$ instead. The proof is called *multiple* simulation-sound if $\mathsf{K}$ can still extract a witness from a new accepted pair $(x^*, \pi^*)$, even if adversary $\mathcal{A}$ can see several proofs $\pi_i$ generated by $\mathsf{Z}_1$ for adaptively chosen statements $x_i$ before.

## C Proof of Theorem 4.2 (Non-Malleability of Our Construction)

To prove non-malleability we present a simulator $\mathcal{S}$ that, with black-box access to any adversary $\mathcal{A}$, manages to succeed in the non-malleability experiment $\mathbf{Exp}^{\text{nmh-0}}_{\mathcal{H},\mathcal{S}}(k)$ almost as often as $\mathcal{A}$ does in $\mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}}(k)$.

The simulator $\mathcal{S}$ works as follows. In mode $\mathcal{S}_d$ it gets as input a key $K_0 = (K_{\mathsf{pow}}, \mathrm{crs}_0)$ of our hash function and invokes the zero-knowledge simulator to generate $(\mathrm{crs}, \sigma) \overset{\$}{\leftarrow} \mathsf{Z}_0(1^k)$. It substitutes $\mathrm{crs}_0$ in $K_0$ by $\mathrm{crs}$ to get $K = (K_{\mathsf{pow}}, \mathrm{crs})$ and runs $\mathcal{A}_d(K)$ to get $(\mathcal{X}, \mathrm{st}_d)$ and outputs $\mathcal{X}$ and $(K, \sigma, \mathrm{st}_d)$ as state. In the next stage $\mathcal{S}_x$ receives side information $h_x$ (and the state). It first picks $x' \overset{\$}{\leftarrow} \mathcal{X}(1^k)$ and computes $y'_{\mathsf{pow}} \overset{\$}{\leftarrow} \mathsf{POW}(K_{\mathsf{pow}}, x'; r')$ for random $r'$. It also invokes the zero-knowledge simulator to generate a simulated proof $\pi' \overset{\$}{\leftarrow} \mathsf{Z}_1(\sigma, K_{\mathsf{pow}}||y'_{\mathsf{pow}})$. Let $y' = (y'_{\mathsf{pow}}, \pi')$ and run $\mathcal{A}_y$ on input $(y', h_x, \mathrm{st}_d)$ to get an output $(y^*, \mathrm{st}_y)$. Then compute a witness through the knowledge extractor, $x^*_{\mathsf{K}}||r^*_{\mathsf{K}} \overset{\$}{\leftarrow} \mathsf{K}(\sigma, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*)$, and return the value $x^*_{\mathsf{K}}$.

To analyze the simulator's behavior we consider a sequence of games in which we run the adversary $\mathcal{A}$ on varying inputs. We denote by $\mathsf{Game}_0(k)$ the original attack of the adversary and $\mathsf{Game}_3(k)$ corresponds to the simulator's experiment. We show that each transition from $\mathsf{Game}_i(k)$ to $\mathsf{Game}_{i+1}(k)$ does not decrease the output distribution noticeably. Hence, the success probability in the simulator's experiment is at least as large as the one in the original attack of the adversary (minus a negligible amount), proving our claim. The games are described formally in Figure C on page 26.

COMPARING GAMES ZERO AND ONE. In comparison to the original attack in $\mathsf{Game}_0(k)$, the modified game $\mathsf{Game}_1(k)$ deploys the zero-knowledge simulator to prepare the common reference string and a fake proof, and replaces the requirement $(x, y) \neq (x^*, y^*)$ simply by $y \neq y^*$. The latter step cannot increase the adversary's success probability by more than a negligible amount, otherwise it is straightforward to derive a successful collision-finder for the POWHF (running the original experiment and using the adversary as a subroutine to get $x^* \neq x$ with the same hash value $y = y^*$).

We next show that the outputs of both games are indistinguishable if $\Pi$ is zero-knowledge. Namely, assume towards contradiction that the probabilities of returning 1 in experiments $\mathsf{Game}_0(k)$ and $\mathsf{Game}_1(k)$ differ noticeably. We then construct an algorithm $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ which is able to distinguish between genuine and simulated proofs for $\Pi$.

Algorithm $\mathcal{D}_0$ gets as input $1^k$ and a string $\mathrm{crs}$, either generated by $\mathsf{CRS}(1^k)$ or as part of the output $(\mathrm{crs}, \sigma)$ of $\mathsf{Z}_0(1^k)$. It next computes a POWHF value $y$ by sampling $K_{\mathsf{pow}} \overset{\$}{\leftarrow} \mathsf{POWK}(1^k)$ and $x \overset{\$}{\leftarrow} \mathcal{X}(1^k)$ for $(\mathcal{X}, \mathrm{st}_d) \overset{\$}{\leftarrow} \mathcal{A}_d(K)$ for $K = (K_{\mathsf{pow}}, \mathrm{crs})$. It also computes $h_x \overset{\$}{\leftarrow} \mathsf{hint}(K, x)$, $r \overset{\$}{\leftarrow} \mathrm{RND}_{K_{\mathsf{pow}}}$ and computing $y_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x; r)$. It outputs $(K_{\mathsf{pow}}||y, x||r)$ for which a proof $\pi$ is generated, either produced by $\mathsf{P}(\mathrm{crs}, K_{\mathsf{pow}}||y_{\mathsf{pow}}, x||r)$ or by the zero-knowledge simulator $\mathsf{Z}_1(\sigma, K_{\mathsf{pow}}||y_{\mathsf{pow}})$. Algorithm $\mathcal{D}_1$ next invokes $\mathcal{A}_y$ on input $(y, h_x, \mathrm{st}_d)$ for $y = (y_{\mathsf{pow}}, \pi)$ to receive $(y^*, \mathrm{st}_y)$ where $y^* = (y^*_{\mathsf{pow}}, \pi^*)$. Algorithm $\mathcal{D}_1$ computes $x^* \overset{\$}{\leftarrow} \mathcal{A}_x(x, \mathrm{st}_y)$ and outputs 1 if and only if $y \neq y^*$, $R(\mathcal{X}, x, x^*) = 1$, $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$ and $\mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1$.

It is easy to see that $\mathcal{D}_1$ returns 1 with the same probability as $\mathsf{Game}_0(k)$ (with the modified check $y \neq y^*$) if the string $\mathrm{crs}$ and the proof $\pi$ are generated by $\mathsf{CRS}$ and $\mathsf{P}$. If, on the other hand, $\mathrm{crs}$ and $\pi$ are created by the simulator $\mathsf{Z}$, then $\mathcal{D}$ outputs 1 with the same probability as $\mathsf{Game}_1(k)$ returns 1. Hence, if both probabilities for the games would differ noticeably, then we would derive a successful distinguisher against the zero-knowledge property.

Comparing Games One and Two. In $\mathsf{Game}_2(k)$, instead of computing $x^* \stackrel{\$}{\leftarrow} \mathcal{A}_x(x, \mathrm{st}_y)$ only, we also run the knowledge extractor if $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$ to derive a hash function pre-image $x^*_{\mathsf{K}}||r^*_{\mathsf{K}} \stackrel{\$}{\leftarrow} \mathsf{K}(\sigma, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*)$ from $\mathcal{A}_y$'s output (else, if $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 0$, we set $x^*_{\mathsf{K}}||r^*_{\mathsf{K}} \leftarrow \perp$ to force the output of the game to be $0$). Experiment $\mathsf{Game}_2(k)$ then uses the extracted value $x^*_{\mathsf{K}}$ to evaluate the decision and outputs 1 iff $y \neq y^*$, $R(\mathcal{X}, x, x^*_{\mathsf{K}}) = 1$, $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 1$ and $\mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1$.

It will follow from the collision-resistance of $\mathcal{P}$ and the simulation soundness of $\Pi$ that the output of these experiments cannot differ significantly. To compare the probabilities we consider the probability of $x^* \neq x^*_{\mathsf{K}}$ with respect to the following disjoint events, conditioning in all cases implicitly on event

$$\mathsf{valid} \;\equiv\; \left[\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1 \;\wedge\; \mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1 \;\wedge\; y \neq y^*\right],$$

which is necessary for output 1 in both games:

- $x^* \neq x^*_{\mathsf{K}}$ and $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 0$
  In this case it would particularly hold $\mathsf{POW}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}; r^*_{\mathsf{K}}) \neq y^*_{\mathsf{pow}}$ (else the verification would succeed by the completeness of the perfectly one-way hash function). But then, since we additionally have $\mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1$ and $y \neq y^*$, the probability for this must be negligible, else it would be straightforward to construct a successful attack against the simulation soundness of the proof system (yielding a valid proof but for which the extractor returns an invalid witness $x^*_{\mathsf{K}}||r^*_{\mathsf{K}}$ not mapping to the hash value $y^*_{\mathsf{pow}}$).

- $x^* \neq x^*_{\mathsf{K}}$ and $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 1$ and $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$
  If this would happen with noticeable probability it would straightforwardly contradict the collision-resistance of the perfectly one-way hash function.

- $x^* \neq x^*_{\mathsf{K}}$ and $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 1$ and $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 0$
  This case cannot happen since we set $x^*_{\mathsf{K}} \leftarrow \perp$ if the verification of the adversarial pre-image $x^*$ fails.

Hence, we have $\Pr\left[\, x^* = x^*_{\mathsf{K}} \,\right]$ with probability negligibly close to 1 (conditioning on $\mathsf{valid}$). The fact $x^* = x^*_{\mathsf{K}}$, on the other hand, guarantees that $\mathsf{Game}_2(k)$ returns 1 with the same probability as $\mathsf{Game}_1(k)$. It follows for some negligible function $\nu(k)$ that

$$\begin{aligned}
&\Pr\left[\, \mathsf{Game}_2(k) = 1 \,\right] \\
&\qquad = \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\wedge\; \neg\mathsf{valid} \,\right] + \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\wedge\; \mathsf{valid} \,\right] \\
&\qquad = \Pr\left[\, \neg\mathsf{valid} \,\right] \cdot \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\middle|\; \neg\mathsf{valid} \,\right] + \Pr\left[\, \mathsf{valid} \,\right] \cdot \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\middle|\; \mathsf{valid} \,\right] \\
&\qquad = 0 + \Pr\left[\, \mathsf{valid} \,\right] \cdot (\Pr\left[\, x^* = x^*_{\mathsf{K}} \;\middle|\; \mathsf{valid} \,\right] \cdot \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\middle|\; x^* = x^*_{\mathsf{K}}, \mathsf{valid} \,\right] \\
&\qquad\qquad\qquad + \Pr\left[\, x^* \neq x^*_{\mathsf{K}} \;\middle|\; \mathsf{valid} \,\right] \cdot \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\middle|\; x^* \neq x^*_{\mathsf{K}}, \mathsf{valid} \,\right]) \\
&\qquad = \Pr\left[\, \mathsf{valid} \,\right] \cdot ((1 - \nu(k)) \cdot \Pr\left[\, \mathsf{Game}_1(k) = 1 \;\middle|\; \mathsf{valid} \,\right] \\
&\qquad\qquad + \nu(k) \cdot \Pr\left[\, \mathsf{Game}_2(k) = 1 \;\middle|\; x^* \neq x^*_{\mathsf{K}}, \mathsf{valid} \,\right]) \geq \Pr\left[\, \mathsf{Game}_1(k) = 1 \,\right] - \nu(k)
\end{aligned}$$

and the probabilities of successful runs in $\mathsf{Game}_2(k)$ can only increase (except for a negligible amount).

Comparing Games Two and Three. The transition to $\mathsf{Game}_3(k)$ consists of two modifications. First, instead of checking that $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$ and running the extractor only then, we now extract $x^*_{\mathsf{K}}$ in any case. This can only increase the success probability of the experiment. As for the second modification we choose an independent $x' \stackrel{\$}{\leftarrow} \mathcal{X}(1^k)$ and compute the hash value as $y_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x'; r')$ for $r' \stackrel{\$}{\leftarrow} \mathrm{RND}_{K_{\mathsf{pow}}}$, whereas we still measure the extracted value $x^*_{\mathsf{K}}$ with respect to the original value $x$. It follows from the perfect one-wayness of the hash function that the probabilities of generating output 1 in experiments $\mathsf{Game}_2(k)$ and $\mathsf{Game}_3(k)$ cannot be affected by this modification noticeably. Assume towards contradiction that this was not the case, and fix a "bad" relation $R \in \mathcal{R}^{\mathsf{rinfo}}_{\mathsf{pred}}$ given by predicates $P_{\mathsf{pow}}$, $P$ and $P^*$ (and by $\mathsf{rinfo}$).

Consider the adversary $\mathcal{B}_d$ which gets as input $K_{\mathsf{pow}}$. Algorithm $\mathcal{B}_d$ runs the zero-knowledge simulator to create $(\mathrm{crs}, \sigma)$ and invokes $\mathcal{A}_d$ on $K = (K_{\mathsf{pow}}, \mathrm{crs})$ to create a distribution $\mathcal{X}$ (and state $\mathrm{st}_d$). Algorithm $\mathcal{B}_d$ outputs $\mathcal{X}$ and state $(\mathrm{st}_d, K, \sigma)$. In the next stage $\mathcal{B}_b$ receives an image $y_{\mathsf{pow}}$ of $(x, r)$ or of $(x', r')$, together with auxiliary information $(r_x, h_x) \overset{\$}{\leftarrow} \mathsf{aux}(K, x) = (\mathsf{rinfo}(x), \mathsf{hint}(K, x))$. Algorithm $\mathcal{B}_b$ runs the zero-knowledge simulator to create a simulated proof $\pi$ for $K_{\mathsf{pow}}||y_{\mathsf{pow}}$. It next invokes a black-box simulation of $\mathcal{A}_y$ on input $((y_{\mathsf{pow}}, \pi), h_x, \mathrm{st}_d)$ to derive $(y^*, \mathrm{st}_y)$. Given these values $\mathcal{B}_b$ extracts a pre-image $x^*_{\mathsf{K}}$ of $y^*$ with help of the knowledge extractor and $\sigma$. It finally outputs $b \leftarrow P^*(r_x, x^*_{\mathsf{K}})$ iff $y \neq y^*$, $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*) = 1$ and $\mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1$; if any of the conditions is violated then $\mathcal{B}$ returns a random bit.

Note that our goal is to show that the outputs $(K, x, b)$ in the two cases (for $x$ and $x'$) are indistinguishable. The analysis is a bit involved, due to the fact that we check for $\mathcal{A}$'s success when switching from $x$ to $x'$ in two stages (some checks performed by $\mathcal{B}$ and the check $P(x, b)$ basically performed by the final distinguisher). To show the claim first let $\mathsf{cond}$ be the event that the conditions $y \neq y^*$, $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*) = 1$ and $\mathsf{V}(\mathrm{crs}, K_{\mathsf{pow}}||y^*_{\mathsf{pow}}, \pi^*) = 1$ *and* $P_{pow}(\mathcal{X}) = 1$ for $\mathcal{A}$'s output are satisfied (given either an image of $x$ or of $x'$). Then it is easy to see that $\Pr[\,\mathsf{cond} \mid x\,]$ is negligibly close to $\Pr[\,\mathsf{cond} \mid x'\,]$, else one could easily devise an attacker against the POWHF which merely checks for these conditions and outputs $b' = 1$ if and only if they are satisified (i.e., such that the tuples $(K, x, b')$ would be easily distinguishable). So from now on we condition on both probabilities being negligibly close.

Given $K, x$ and $\mathcal{B}$'s output bit $b$ it is now easy to distinguish the two cases for an algorithm $\mathcal{D}$ by simply verifying that $P(x, b) = 1$. Note that, given $\neg\mathsf{cond}$ and $\mathcal{B}$ has returned a random bit $b$, the probabilities for $P(x, b) = 1$ are identical for both cases $x$ and $x'$. Hence,

$$\begin{aligned}
&\Pr[\,\mathcal{D}(K, x, b) = 1 \mid x\,] \\
&\quad = \Pr[\,\mathcal{D}(K, x, b) = 1 \wedge \mathsf{cond} \mid x\,] + \Pr[\,\mathcal{D}(K, x, b) = 1 \wedge \neg\mathsf{cond} \mid x\,] \\
&\quad = \Pr[\,P(x, b) = 1 \wedge \mathsf{cond} \mid x\,] + \Pr[\,P(x, b) = 1 \wedge \neg\mathsf{cond} \mid x\,] \\
&\quad = \Pr[\,P(x, b) = 1 \wedge \mathsf{cond} \mid x\,] + \Pr[\,P(x, b) = 1 \mid \neg\mathsf{cond}, x\,] \cdot \Pr[\,\neg\mathsf{cond} \mid x\,]
\end{aligned}$$

and analogously for the case $x'$:

$$\begin{aligned}
&\Pr\left[\,\mathcal{D}(K, x, b) = 1 \mid x'\,\right] \\
&\quad = \Pr\left[\,P(x, b) = 1 \wedge \mathsf{cond} \mid x'\,\right] + \Pr\left[\,P(x, b) = 1 \mid \neg\mathsf{cond}, x'\,\right] \cdot \Pr\left[\,\neg\mathsf{cond} \mid x'\,\right]
\end{aligned}$$

Recall that the probabilities $\Pr[\,\neg\mathsf{cond} \mid x\,]$ and $\Pr[\,\neg\mathsf{cond} \mid x'\,]$ are negligibly close (as discussed above), and that the conditional probabilities for $P(x, b) = 1$ (given $\neg\mathsf{cond}$ and $x$ resp. $x'$) are identical. Hence, the products are therefore negligibly close, too. The other two probabilities $\Pr[\,P(x, b) = 1 \wedge \mathsf{cond} \mid x\,]$ and $\Pr[\,P(x, b) = 1 \wedge \mathsf{cond} \mid x'\,]$ correspond to the cases that $\mathcal{A}$ suceeds in the two games. Since they have a non-negligible difference by assumption this contradicts the perfect one-wayness of $\mathcal{P}$.

The final game now mirrors the simulator's strategy and the corresponding experiment (except that the simulator does not need to obey the stipulations $y \neq y^*$ and $\mathsf{HVf}(K, x^*, y^*) = 1$, which can only increase its success probability further). This proves non-malleability.

COLLISION-RESISTANCE. It remains to show that the hash function is collision-resistant. But this follows straightforwardly from the collision-resistance of the perfectly one-way hash function. ▌

# D Auxiliary Results for Our Black-Box Separation

This section covers some formal statements and proofs for the black-box separation result.

BLACK-BOX REDUCTIONS. We first recall the more formal definition of black-box reductions from [21]:

**Definition D.1 (Black-Box Reduction)** *A* black-box reduction *from a non-malleable hash function (with respect to some* $\mathsf{hint}^?$*) to one-way permutations consists of efficient algorithms* $\mathcal{H}^? = (\mathsf{HK}^?, \mathsf{H}^?, \mathsf{HVf}^?)$ *and* $\mathcal{A}^?_{\mathcal{P}}$ *with the following properties. For all algorithms* $\mathcal{P}$ *and* $\mathcal{A}_{\mathcal{H}}$*, each of arbitrary complexity,*

- *Correctness: If $\mathcal{P}$ is a permutation (i.e., is a bijection over $\{0,1\}^k$ for each $k \in \mathbb{N}$), then $\mathcal{H}^{\mathcal{P}}$ is a hash function.*
- *Security: If $\mathcal{A}_{\mathcal{H}}$ breaks the non-malleability of the hash function $\mathcal{H}^{\mathcal{P}}$ with respect to $\mathsf{hint}^{\mathcal{P}}$ (i.e., violates Definition 3.1), then $\mathcal{A}_{\mathcal{P}}^{\mathcal{A}_{\mathcal{H}}}$ breaks the one-wayness of $\mathcal{P}$.*

We use the following lemma of [21] (adapted to our setting), which generalizes the standard technique of proving the impossibility of a black-box reduction by giving oracles relative to which the reduction cannot exist.

**Lemma D.2 (adapted from [21])** *No black-box reduction from non-malleable hash functions (with respect to some $\mathsf{hint}^?$) can exist if there exists oracles $\mathcal{P}$ and $\mathcal{O}$ with the following properties:*

- *$\mathcal{P}$ is a one-way permutation secure against all PPT $\mathcal{B}^{\mathcal{P},\mathcal{O}}$.*
- *For all PPT $\mathcal{H}^? = (\mathsf{HK}^?, \mathsf{H}^?, \mathsf{HVf}^?)$, there exists a PPT adversary $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ that breaks non-malleability of $\mathcal{H}^{\mathcal{P}}$ (with respect to some $\mathsf{hint}^{\mathcal{P}}$)*

Note that breaking non-malleability means that no simulator of comparable complexity is able to approximate the success probability of $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ close enough. To ensure that the simulator has equal power we grant the simulator $\mathcal{S}^{\mathcal{P},\mathcal{O}}$ therefore access to both oracles $\mathcal{P}, \mathcal{O}$.

DEFINING ORACLES $\mathcal{P}$ AND $\mathcal{O}$. Recall that we let $\mathcal{P}$ be a random permutation (which we will later derandomize) and that our oracle $\mathcal{O}$ on input $(1^k, y, p)$ looks up the pre-image $0^k||x||\langle\mathsf{HVf}\rangle||K$ of $h_x$ under $\mathcal{P}$ and returns $x$ if and only if $z = 0^k$ and $\mathsf{HVf}^{\mathcal{P}}(K, x, y) = 1$. We first show that $\mathcal{O}$ does not help to break the one-wayness of $\mathcal{P}$:

**Proposition D.3** *For any efficient algorithm $\mathcal{B}^{?,?}$, the probability that $\mathcal{B}^{\mathcal{P},\mathcal{O}}$ breaks the one-wayness of $\mathcal{P}$ is negligible.*

**Proof:** Assume that there exist an efficient algorithm $\mathcal{B}^{\mathcal{P},\mathcal{O}}$ that breaks the one-wayness with noticeable probability $1/q(k)$ for a polynomial $q$ and infinitely many $k$'s. Let $t(k)$ be the polynomial bounding the running time of $\mathcal{B}$. Then we construct an efficient algorithm $\mathcal{C}^{\mathcal{P}}$ which inverts images with almost the same success probability, but without the help of $\mathcal{O}$.

Algorithm $\mathcal{C}^{\mathcal{P}}$ is given some $\beta \in \{0,1\}^k$ as input and first queries its oracle $\mathcal{P}$ for all values of size $B := B(k) := \log_2(2q(k)t(k) + t(k))$. It records all those queries and answers in a list $L_{\mathcal{P}}$, which is possible in polynomial space and time. It then starts a black-box simulation of $\mathcal{B}^{\mathcal{P},\mathcal{O}}(\beta)$ in which $\mathcal{C}$ answers each query of $\mathcal{B}$ to $\mathcal{P}$ with the help of oracle $\mathcal{P}$, but where $\mathcal{C}$ also appends all queries and answers to the list $L_{\mathcal{P}}$.

Each of $\mathcal{B}$'s queries $(1^{k'}, y, p)$ to oracle $\mathcal{O}$ is processed by $\mathcal{C}$ as follows. $\mathcal{C}$ searches through the list $L_{\mathcal{P}}$ so far and checks whether it has stored a pair in which $p$ appears as the image. If so then $\mathcal{C}$ proceeds as the oracle $\mathcal{O}$ would (i.e., checks that the leading bits of the pre-image are zero and that the final part verifies with respect to $y$, and returns the pre-image if all tests succeed). If, on the other hand, no such entry exists in $L_{\mathcal{P}}$ then $\mathcal{C}$ simply returns $\perp$. Algorithm $\mathcal{C}$ finally outputs whatever $\mathcal{B}$ returns.

For the analysis consider the $i$-th query $(1^{k'}, y, p)$ of $\mathcal{B}$ to $\mathcal{O}$, conditioning on the fact that $\mathcal{C}$ has answered all $i-1$ previous queries consistently with $\mathcal{O}$'s replies. If $\mathcal{C}$ finds a corresponding value/image pair in the list $L_{\mathcal{P}}$ then the reply for this reply is also consistent with $\mathcal{O}$'s answer. Assume that there is no such pair in $L_{\mathcal{P}}$. In particular, the parameter $k'$ must then be larger than $B$. At any point during the simulation $\mathcal{B}$ has gathered at most $t(k)$ value/image pairs for $\mathcal{P}$ (where we also count the information derived through previous $\mathcal{O}$ queries, possibly showing only that the leading bits of these pre-images are not zero). Hence, the probability that for the $k'$ most significant bits $\mathrm{msb}_{k'}(\mathcal{P}^{-1}(p)) = 0^{k'}$ is at most $\frac{1}{2^{k'}-t(k)}$, and thus at most $\frac{1}{2q(k)t(k)}$. Summing over all at most $t(k)$ queries of $\mathcal{B}$ to $\mathcal{O}$ all answers of $\mathcal{C}$ are consistent with probability at least $1 - \frac{1}{2q(k)}$.

In conclusion, the probability that $\mathcal{C}$ succeeds is bounded from below by the probability $\frac{1}{2q(k)}$. In addition, $\mathcal{C}$ runs in polynomial time in $k$ and thus contradicts the one-wayness of $\mathcal{P}$. ▮

SEPARATION. Recall that we need the technical assumption that the hash function is non-trivial in the sense that for uniformly distributed input the output has enough min-entropy. For sake of simplicity we state this requirement without the oracles $\mathcal{P}, \mathcal{O}$:

**Definition D.4** *A hash function $\mathcal{H}$ is non-trivial if, for any $K \leftarrow \mathsf{HK}(1^k)$ and any $y$, the probability that $\mathsf{HVf}(K, x, y) = 1$ for $x \xleftarrow{\$} \{0,1\}^k$, is negligible.*

As another technical prerequisite we assume that the relation $R_{\mathsf{sep}}(\mathcal{X}, x, x^*)$ which checks that $\mathcal{X}$ is the uniform distribution on $\{0,1\}^k$ and that the parity of $x$ and $x^*$ are equal, is in the relation class $\mathcal{R}$. Note that clearly $R_{\mathsf{sep}} \in \mathcal{R}_{\text{pred}}$ is in the class of relations for which our construction holds.

With the properties above we can show the following:

**Theorem D.5** *Let $\mathcal{H}^{\mathcal{P}} = (\mathsf{HK}^{\mathcal{P}}, \mathsf{H}^{\mathcal{P}}, \mathsf{HVf}^{\mathcal{P}})$ be a non-trivial non-malleable hash function with respect to $\mathsf{hint}^{\mathcal{P}}_{\mathsf{sep}}$ and $\mathcal{R} \ni R_{\mathsf{sep}}$. Then there exists an adversary $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ that breaks non-malleability of $\mathcal{H}^{\mathcal{P}}$ (for any simulator $\mathcal{S}^{\mathcal{P},\mathcal{O}}$).*

**Proof:** Consider the adversary $\mathcal{A}_d^{\mathcal{P},\mathcal{O}}$ which for input $K$ returns (a description of) the uniform distribution $\mathcal{X}$ over $\{0,1\}^k$ and the state $\mathrm{st}_d = K$. Adversary $\mathcal{A}_y$ on input $y, h_x$ and $\mathrm{st}_d$ forwards $(1^k, y, h_x)$ to oracle $\mathcal{O}$ to recover $x$. It then continously samples $x^* \xleftarrow{\$} \mathcal{X}(1^k)$ till it finds one with equal parity as $x$ (or stops after at most $n$ trials with undefinded output). It computes a hash value $y^* \leftarrow \mathsf{H}^{\mathcal{P}}(K, x^*)$ and returns this value (together with the state $\mathrm{st}_y = x^*$). Adversary $\mathcal{A}_x$ for input $x$ and $\mathrm{st}_y = x^*$ simply outputs $x^*$. Note that $\mathcal{A}$'s output satisfies relation $R_{\mathsf{sep}}$ and $(x, y) \neq (x^*, y^*)$ with probability negligibly close to 1 (there is a small probability of $2^{-n}$ that $\mathcal{A}_y$ cannot find a suitable $x^*$ within $n$ trials, and a negligible probability that $x^* = x$).

Consider now an arbitrary (efficient) simulator $\mathcal{S}_x^{\mathcal{P},\mathcal{O}}$, making at most $t(k)$ queries to either oracle for polynomial $t(k)$. First note that the distribution $\mathcal{X}$ output by $\mathcal{S}_d^{\mathcal{P},\mathcal{O}}(K)$ must also be the uniform distribution with overwhelming probability, and from now on we condition on this. Consider both stages, $\mathcal{S}_d$ and $\mathcal{S}_x$, where the latter algorithm gets $h_x = \mathcal{P}(0^k||x||\langle\mathsf{HVf}\rangle||K)$ and $\mathrm{st}_d$ as input. We first claim that the probability that $\mathcal{S}$ receives from $\mathcal{O}$ the pre-image $x$ or puts a query to $\mathcal{P}$ including $x$ in any of the two stages (event $\mathsf{BAD}$) cannot be more than negligible.

Consider the $(i+1)$-st query which is either a $\mathcal{P}$-query or a query to $\mathcal{O}$, assuming that none of the previous $i$ queries has triggered event $\mathsf{BAD}$. If this is a query to oracle $\mathcal{P}$ then $\mathcal{S}$ has gathered information about at most $i$ other $\mathcal{P}$-values so far (either directly or through $\mathcal{O}$), thus finding the unique pre-image and causing event $\mathsf{BAD}$ with probability at most $\frac{1}{2^k - i}$ (which is negligible for large $k$'s). Now suppose that the $i$-th query $(1^{k'}, y', p)$ is to oracle $\mathcal{O}$. If $p = h_x$ then the answer can only be $x$ if $\mathsf{HVf}(K, x, y') = 1$. But this can only happen with negligible probability, as the only information about $x$ at this point is that it is different from all previously seen pre-images under $\mathcal{P}$, and the non-triviality of the hash function implies that the verification succeeds with negligible probability only. In case $p \neq h_x$ the probability that $\mathcal{O}$ returns $x$ is at most $\frac{1}{2^k - i}$, because the pre-image of $p$ under $\mathcal{P}$ is either known by $\mathcal{S}$ via a previous query and therefore distinct from $x$, or the probability that the pre-image under $\mathcal{P}$ contains $x$ is at most $\frac{1}{2^k - i}$.

It follows that, for any of the at most polynomially many queries to its oracles, $\mathcal{S}$ receives a useful answer causing even $\mathsf{BAD}$ with negligible probability only. Hence, with overwhelming probability any of the $2^k - t(n)$ possible pre-images of $h_x$ under $\mathcal{P}$ is still equally like from the viewpoint of $\mathcal{S}$, meaning that the simulator cannot approximate the parity of $x$ better than with probability negligibly close to $\frac{1}{2}$, which is noticeably away from $\mathcal{A}$'s success probability. ▮

It now follows easily:

**Corollary D.6** *There exists no black-box reduction from non-trivial non-malleable functions (with respect to* $\mathsf{hint}^{\mathcal{P}}_{\mathit{sep}}$ *and* $\mathcal{R} \ni R_{\mathit{sep}}$*) to one-way permutations.*

# E Definition and Proof for Proposition 5.5 (NM Implies POWHFs)

We first recall the oracle-simulatability definition of perfect one-wayness from [8] for adversaries that get auxillary information. This version of the definition is slightly different from the original in how handle message distributions. Here, we allow the adversary and simulator to pick the message distribution after seeing the hash key instead of quantifying over all message distributions. We also allow the experiment to run a predicate on the chosen message distribution before declaring if the adversary has won.

**Definition E.1 (POWHF, oracle-simulatability definition)** *Let* $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ *be a hash function. It is* perfectly one-way in the oracle-simulatability sense *with respect to a function* $\mathsf{hint}$ *and predicate* $P_{\mathsf{pow}}$*, if if for any PPTA* $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_p)$ *and any PT predicate* $\Pi_x$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_p)$ *such that the difference*

$$\Pr\left[\mathbf{Exp}^{\mathrm{pow}-1}_{\mathcal{H},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{pow}-0}_{\mathcal{H},\mathcal{S}}(k) = 1\right]$$

*is negligible, where:*

```
***Experiment*** $\mathbf{Exp}^{\mathrm{pow}-1}_{\mathcal{H},\mathcal{A}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, \mathsf{st}_d) \xleftarrow{\$} \mathcal{A}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k),\ \ h_x \leftarrow \mathsf{hint}(x)$
    $y \leftarrow \mathsf{H}(K, x)\ ;\ p \leftarrow \mathcal{A}_p(y, h_x, \mathsf{st}_d)$
    *Return 1 iff* $p = \Pi_x(x) \wedge P_{\mathsf{pow}}(\mathcal{X}) = 1$
```

```
***Experiment*** $\mathbf{Exp}^{\mathrm{pow}-0}_{\mathcal{H},\mathcal{S}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$
    $(\mathcal{X}, \mathsf{st}_d) \xleftarrow{\$} \mathcal{S}_d(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k),\ \ h_x \leftarrow \mathsf{hint}(x)$
    $p \leftarrow \mathcal{S}_p(h_x, \mathsf{st}_d)$
    *Return 1 iff* $p = \Pi_x(x) \wedge P_{\mathsf{pow}}(\mathcal{X}) = 1$
```

*We only consider predicates* $P_{\mathsf{pow}}$ *for which* $P_{\mathsf{pow}}(\mathcal{X}) = 1$ *implies that* $\mathcal{X}$ *is well-spread.*

REMARK 1. Because the definition in [8] considers arbitrary distributions, the simulator in the "ideal" experiment is given the oracle that verifies whether a given pre-image is the one chosen by the experiment. Since a perfectly one-way hash function should usually be at least one-way, we consider only well-spread distributions (therefore the restriction that we put on $P_{\mathsf{pow}}$) and hence such an oracle is of no use to the simulator.

REMARK 2. The introduction in [8] suggests that a perfectly one-way hash function must be randomized, and all the suggested constructions are. However, a deterministic hash function does not violate perfect one-wayness, and the constructions in [8, 11] are still POWHF if the randomness becomes part of the key and are used for a single message. Hence, our implication that non-malleability implies perfect one-wayness does not necessarily mean that non-malleable hash functions must be probabilistic.

**Proof of Proposition 5.5:** The goal is to show that $\mathcal{H}$ is perfectly one-way with respect to $\mathsf{hint}$ and $P_{\mathsf{pow}}$. Fix some adversary $\mathcal{A}_{\text{pow}}$ that attacks perfect one-wayness according to the above definition for some predicate $\Pi_x$. We construct an adversary $\mathcal{A}_{\text{nm}}$ and a relation $R \in \mathcal{R}^{P_{\mathsf{pow}}}_{\text{pred}}$ so that $\mathcal{A}_{\text{nm}}$ performs almost as well in the non-malleability experiment as $\mathcal{A}_{\text{pow}}$ does in the oracle-simulatability experiment. Then, we use the non-malleability of $\mathcal{H}$ to build a simulator $\mathcal{S}_{\text{pow}}$ that performs almost as well as $\mathcal{A}_{\text{pow}}$.

Below we let $x_0$ be the string output by fixing the random tape of the sampling algorithm $\mathcal{X}(1^k)$ to all zeros. (We simply need $x_0$ to be an element of the message space that $\mathcal{X}(1^k)$ outputs with low probability.)

Now we can describe the NM adversary $\mathcal{A}_{\text{nm}} = (\mathcal{A}_{\text{nm},d}, \mathcal{A}_{\text{nm},y}, \mathcal{A}_{\text{nm},x})$. The first algorithm, $\mathcal{A}_{\text{nm},d}$, is almost exactly like $\mathcal{A}_{\text{pow},d}$; it outputs the same $\mathcal{X}$ and $\text{st}_d$, except that it appends the hash key $K$ to $\text{st}_d$. The third algorithm, $\mathcal{A}_{\text{nm},x}$, simply outputs $\text{st}_y$. To complete the description of our NM adversary, and the relation it attacks, $\mathcal{A}_{\text{nm},y}$ and $R$ are as follows:

```
**Adversary** $\mathcal{A}_{\text{nm},y}(y, h_x, \text{st}_d)$
    $p \leftarrow \mathcal{A}_{\text{pow},p}(y, h_x, \text{st}_d)$
    if $p = 0$ then $x^* \leftarrow x_0$ else $x^* \xleftarrow{\$} \mathcal{X}(1^k)$
    $y^* \leftarrow \mathsf{H}(K, x^*)$ ; $\text{st}_y \leftarrow x^*$
    output $(y^*, \text{st}_y)$
```

```
**Relation** $R(x, x^*, \mathcal{X})$
    output $(\Pi_x(x) \oplus [x^* = x_0]) \wedge P_{\mathsf{pow}}(\mathcal{X})$
```

Clearly, $R$ belongs to the class $\mathcal{R}^{P_{\mathsf{pow}}}_{\text{pred}}$ because it can be expressed as $P(x, P^*(x^*))$ by setting $P^*(x^*) = [x^* = x_0]$ and $P(x, b) = \Pi(x) \oplus b$. By writing out and re-arranging $\mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}_{\text{nm}}}$, we get:

```
**Experiment** $\mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}_{\text{nm}}}(k)$
    $K \xleftarrow{\$} \mathsf{HK}(1^k)$ ; $\mathcal{X}(1^k) \leftarrow \mathcal{A}_{\text{pow},d}(K)$
    $x \xleftarrow{\$} \mathcal{X}(1^k)$ ; $h_x \leftarrow \mathsf{hint}(x)$ ; $y \leftarrow \mathsf{H}(K, x)$
    $p \leftarrow \mathcal{A}_{\text{pow},p}(K, y, h_x)$
    if $p = 0$ then $x^* \leftarrow x_0$ else $x^* \xleftarrow{\$} \mathcal{X}(1^k)$
    $y^* \leftarrow \mathsf{H}(K, x^*)$
    Return 1 iff $(y^* \neq y) \wedge \mathsf{HVf}(K, x^*, y^*) = 1 \wedge (\Pi_x(x) \oplus [x^* = x_0]) \wedge P_{\mathsf{pow}}(\mathcal{X}) = 1$
```

It then follows that the difference $\Pr\left[\, \mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}_{\text{nm}}} \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{pow}-1}_{\mathcal{H},\mathcal{A}_{\text{pow}}}(k) \,\right]$ is negligible. Namely, if $\mathcal{A}_{\text{pow}}$ predicts correctly $p = 0$, then $\mathcal{A}_{\text{nm}}$ also causes $R$ to output 1. If $\mathcal{A}_{\text{pow}}$ predicts correctly $p = 1$, however, there is a chance that the $x^* \neq x_0$ requirement will not be met. But since $\mathcal{X}$ is well-spread, this happens with only negligible probability. The only remaining difference in the outputs of $\mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}_{\text{nm}}}$ could be caused by and $\mathbf{Exp}^{\text{pow}-1}_{\mathcal{H},\mathcal{A}_{\text{pow}}}(k)$ is the $y^* \neq y$, requirement. However, $\Pr\left[\, y^* = y \,\right]$ is negligible since $\mathcal{H}$ is collision-resistant.

By the non-malleability of $\mathcal{H}$ under Definition 3.1, there exists a simulator $\mathcal{S}_{\text{nm}} = (\mathcal{S}_{\text{nm},d}, \mathcal{S}_{\text{nm},x})$ such that the difference $\Pr\left[\, \mathbf{Exp}^{\text{nmh-1}}_{\mathcal{H},\mathcal{A}_{\text{nm}}} = 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{nmh-0}}_{\mathcal{H},\mathcal{S}_{\text{nm}}} = 1 \,\right]$ is negligible. We now complete the proof by constructing a simulator $\mathcal{S}_{\text{pow}}$ (that works analogously to $\mathcal{A}_{\text{nm}}$), and for which we have that $\Pr\left[\, \mathbf{Exp}^{\text{pow}-0}_{\mathcal{H},\mathcal{S}_{\text{pow}}}(k) \,\right] = \Pr\left[\, \mathbf{Exp}^{\text{nmh-0}}_{\mathcal{H},\mathcal{S}_{\text{nm}}} \,\right]$. The simulator $\mathcal{S}_{\text{pow}} = (\mathcal{S}_{\text{pow},d}, \mathcal{S}_{\text{pow},p})$ operates as follows. $\mathcal{S}_{\text{pow},d}$ is exactly $\mathcal{S}_{\text{nm},d}$. $\mathcal{S}_{\text{pow},p}$ is defined by:

```
**Simulator** $\mathcal{S}_{\text{pow},p}(h_x, \text{st}_d)$
    $x^* \leftarrow \mathcal{S}_{\text{nm},x}(h_x, \text{st}_d)$
    if $x^* = x_0$ then $p \leftarrow 0$ else $p \leftarrow 1$
    output $p$
```

Whenever $\mathcal{S}_{\text{nm}}$ causes $R(x, x^*, \mathcal{X}) = 1$, then $\mathcal{S}_{\text{pow}}$ will correctly guess $\Pi_x(x)$ and satisfy $P_{\mathsf{pow}}(\mathcal{X})$. ▮

## F Proof of Proposition 6.1 (IND-CCA Security of BR Encryption)

We first recall the standard definition of security under chosen-ciphertext attack, or IND-CCA security, for public-key encryption.

**Definition F.1 (IND-CCA)** *Let PKE* $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a public-key encryption scheme. It is* IND-CCA secure *if for any PPTA* $\mathcal{B} = (\mathcal{B}_m, \mathcal{B}_b)$*, the difference* $\Pr\left[\, \mathbf{Exp}^{\text{ind-cca}}_{PKE,\mathcal{B}}(k) = 1 \,\right] - 1/2$ *is negligible, where:*

```
***Experiment*** $\mathbf{Exp}^{\text{ind-cca}}_{PKE,\mathcal{B}}(k)$
    $(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$ ; $(m_0, m_1, st) \xleftarrow{\$} \mathcal{B}_m^{\mathcal{D}_{sk}(\cdot)}(pk)$
    $b \xleftarrow{\$} \{0, 1\}$ ; $c \xleftarrow{\$} \mathcal{E}_{pk}(m_b)$
    $b^* \leftarrow \mathcal{B}_b^{\mathcal{D}_{sk}(\cdot)}(c, st)$
    *Return 1 iff* $b^* = b$
```

*We assume that $\mathcal{B}_b$ does not query $\mathcal{D}_{sk}(c)$.*

Let us fix some notation used in the proof. Below we will use the natural modification of this definition where we give all of the algorithms involved access to a random oracle, which we denote $G$. We let $\mathcal{B}$ be an adversary attacking the encryption scheme in the IND-CCA game. We will refer to the challenge ciphertext as $\hat{y}, \hat{g}, \hat{h}$.

We first describe a sequence of games, where $\mathsf{Game}_0$ is equivalent to $\mathbf{Exp}^{\text{ind-cca}}_{\text{PKE},\mathcal{B}}(k)$ from Definition F.1, and the final game is simple to analyze. Next we show that for each pair of adjacent games $\mathsf{Game}_i$ and $\mathsf{Game}_{i+1}$, the adversary's advantage can change only negligibly. In the final game, we complete the proof by showing that the adversary's advantage is negligible.

$\mathsf{Game}_1$. This is like $\mathsf{Game}_0$, except that on decryption query $(y, g, h)$, if $y \neq \hat{y}$ and $\mathcal{B}$ has not queried an $r$ to $G$ such that $y = f(r)$, $\mathsf{Game}_1$ rejects the ciphertext.

$\mathsf{Game}_2$. This game is like $\mathsf{Game}_1$, except now the decryption oracle queries $(y, g, h)$, where $y = \hat{y}$ and $h = \hat{h}$ (and thus $g \neq \hat{g}$), and the adversary has not queried $G(r)$ such that $y = \mathsf{f}(r)$, are rejected.

$\mathsf{Game}_3$. This is like $\mathsf{Game}_2$, except now the decryption oracle rejects *all* ciphertexts $(y, g, h)$ such that $\mathcal{B}$ did not query $G(r)$ such that $y = \mathsf{f}(r)$. (This amounts to adding a new rule to reject ciphertexts with $y = \hat{y}$, $h \neq \hat{h}$, and this property.)

$\mathsf{Game}_4$. This is like $\mathsf{Game}_3$, except that now the challenge ciphertext is set to an encryption of a uniformly chosen random message instead of $m_b$.

COMPARING GAMES ZERO AND ONE Let $F_1$ be the event that $\mathcal{B}$ submits a ciphertext in $\mathsf{Game}_1$ that is rejected, but would not have been rejected in $\mathsf{Game}_0$. It is clear that condition on the event that $F_1$ does not occur, $\mathcal{B}$'s outputs in $\mathsf{Game}_0$ and $\mathsf{Game}_1$ have the same distribution.

We claim that $\Pr[F_1] = \varepsilon_1$ is negligible, by the collision resistance of $\mathcal{H}$. To see this, define the following adversary $\mathcal{A}_1$ that attemps to find collisions in $\mathcal{H}$. Adversary $\mathcal{A}_1$ takes as input a hash key $(K, \mathsf{f})$ and uses it in the public key in $\mathsf{Game}_1$, but replaces $\mathsf{f}$ by sampling an independent instance $\mathsf{f}_0$ (thus knowing the secret key $\mathsf{f}_0^{-1}$) and generates the rest of the public key/secret key pair itself and answers all oracle queries for $\mathcal{B}$, having full control over oracle $G$ (i.e., answering each new query with a new random string and storing all queries and the relies in an array called $G$-list, so the repeated queries could be answered consistently) and also creating the challenge ciphertext as described by the protocol. If $\mathcal{A}_1$ detects a ciphertext $(y, g, h)$ that would get rejected in $\mathsf{Game}_1$, it halts the game. Then adversary $\mathcal{A}_1$ computes $r = \mathsf{f}_0^{-1}(y)$, $m = G(r) \oplus g$, generating the value $G(r)$ from scratch. Attacker $\mathcal{A}_1$ then selects a new random message $m'$, and outputs $(r \| m, r \| m')$ and $h$ as its collision.

Since the ciphertext $(y, g, h)$ would not be rejected in $\mathsf{Game}_0$, we know that $\mathsf{HVf}(K, h, r \| m) = 1$. Observe that when $\mathcal{A}_1$ computed $m = G(r) \oplus g$, it was actually *setting $m$ to a random message*. This is true because $\mathcal{B}$ had never queried $G(r)$, so $\mathcal{A}_1$ selected a random string for it after the game was over. Now observe that an $\varepsilon_1$ fraction of the possible coins that determine $\mathsf{Game}_0$ will cause the event $F_1$ to occur. It then follows by a standard averaging argument that, for an $\varepsilon_1/2$ fraction of the coins for $\mathsf{Game}_0$, there is an $\varepsilon_1/2$ chance over the choice of $G(r)$ (with the other coins fixed) will cause $F_1$. Thus, when the $\mathcal{A}_1$ selects a second message $m'$ after halting the game, there is at least an $\varepsilon_1^2/4$ chance that it will satisfy $\mathsf{HVf}(K, h, r \| m') = 1$. Thus the chance that $\mathcal{A}_1$ finds a collision is at least $\varepsilon_1^3/8$, and $\varepsilon_1$ must negligible.

COMPARING GAMES ONE AND TWO It is clear that until the new reject rule is used, $\mathsf{Game}_1$ and $\mathsf{Game}_2$ are the same. Let $\mathsf{F}_2$ be the event that a query is rejected because of this rule that would not have been rejected in $\mathsf{Game}_1$, and let $\varepsilon_2 = \Pr[F_2]$. We claim that $\varepsilon_2$ is negligible by the collision resistance of $\mathcal{H}$.

We show $\varepsilon_2$ is negligible by presenting an efficient adversary $\mathcal{A}_2$ that takes as input a hash key $K$ finds a collision in $\mathcal{H}$ with probability related to $\varepsilon_2$. Algorithm $\mathcal{A}_2$ also generates a new pair $(\mathsf{f}_0, \mathsf{f}_0^{-1})$ and uses $K$ and $\mathsf{f}_0$ in the public key and runs $\mathsf{Game}_2$ for $\mathcal{B}$ until the reject rule in $\mathsf{Game}_2$ is used (again, controlling $G$ and generating a valid challenge ciphertext). Let $(y, g, h)$ be the ciphertext that triggered the reject rule, so $y = \hat{y}$ and $h = \hat{h}$. Adversary $\mathcal{A}_2$ computes $m' = G(r) \oplus g$, computes $\hat{r} = \mathsf{f}_0^{-1}(\hat{y})$, recalls that $m_b$ was the message used in the challenge ciphertext, and then outputs $(\hat{r} \| m_b, \hat{r} \| m')$ and $\hat{h}$ as its collision.

We claim that $\mathcal{A}_2$ finds a collision with probability $\varepsilon_2$. This is because if the event $F_2$ was used, then the ciphertext $(y, g, h)$ must have been valid, i.e., $\mathsf{HVf}(K, h, \hat{r}\|(G(\hat{r}) \oplus g)) = 1$. By assumption $g \neq \hat{g}$, hence $m' = G(r) \oplus g$ is different from $m_b = G(r) \oplus \hat{g}$ but still maps (prepended by $r = \hat{r}$) to the same hash value $h = \hat{h}$. The claim follows, and $\varepsilon_2$ must be negligible.

COMPARING GAMES TWO AND THREE $\mathsf{Game}_2$ and $\mathsf{Game}_3$ are clearly the same until the new rule is applied. Let $F_3$ be the event that a ciphertext is queried that triggers this reject rule in $\mathsf{Game}_3$. but would not have been rejected in $\mathsf{Game}_2$. We claim that $\Pr[F_3]$ is negligible, by the non-malleability condition on $\mathcal{H}$ in the theorem.

Let $\varepsilon_3 = \Pr[F_3]$. To see this we construct an adversary $\mathcal{A}_3 = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$ to win the non-malleability game with $\mathsf{hint}_{\mathsf{BR}}$ and our relation $R_{\mathsf{BR}} \in \mathcal{R}^{\mathsf{rinfo}}_{\mathsf{pred}^*}$ more often than any simulator. Recall that our relation $R_{\mathsf{BR}}(\mathcal{X}, r_0\|m_0, r_1\|m_1)$ tests if (1) $\mathcal{X}$ is a canonical sampler that samples $r \xleftarrow{\$} \{0,1\}^k$ and outputs $r\|m$ for some $m$ and (2) if the leading $k/2$ bits of $r_0$ and $r_1$ are equal.

Our $\mathcal{A}_d$ takes as input a hash key $K' = (K, \mathsf{f})$ and begins to simulate $\mathsf{Game}_3$ for $\mathcal{B}$ using $(K, \mathsf{f})$ as the public key. To simulate the random oracle $G$ properly, $\mathcal{A}_d$ stores all the random oracle queries $\mathcal{B}$ makes and the corresponding replies in an associative array, the $G$-list. $\mathcal{A}_d$ answers decryption queries $(y, g, h)$ as described in $\mathsf{Game}_2$ by first locating a previously made query $g$ in the $G$-list such that $y = \mathsf{f}(r)$, and rejecting all other ciphertexts. $\mathcal{A}_d$ never accesses $\mathsf{f}^{-1}$ to maintain this simulation.

$\mathcal{A}_d$ runs $\mathcal{B}$ until it outputs two messages $(m_0, m_1)$. Then $\mathcal{A}_d$ chooses a random bit $b$ and outputs the canonical encoding of the distribution $\mathcal{X}$ that selects $r \xleftarrow{\$} \{0,1\}^k$ and outputs $r\|m_b$.

After the expirement samples $\hat{r}\|m_b$ and $\mathcal{A}_y$ receives $\hat{h} \leftarrow H(K, \hat{r}\|m_b), \mathsf{f}(\hat{r})$ as input, $\mathcal{A}_y$ sets $\hat{g}$ to a random string, defines $G(\hat{r}) = \hat{g} \oplus m_b$ and returns $(\hat{y}, \hat{g}, \hat{h})$ as the challenge ciphertext for $\mathcal{B}$. For each of $\mathcal{B}$'s queries $r_i$ to the random oracle $G$, adversary $\mathcal{A}_y$ checks if $\mathsf{f}(r_i) = \hat{y}$, and if so, returns $G(\hat{r})$; else it uses the same lazy sampling technique as before. $\mathcal{A}_y$ continues to answer decryption queries as before until $\mathcal{B}$ halts.

After $\mathcal{B}$ halts, $\mathcal{A}_y$ examines all of the rejected decryption queries issued by $\mathcal{B}$ that had their first part $\hat{y}$ and other part $(g, h) \neq (\hat{g}, \hat{h})$. Out of these $q_{\mathcal{D}}$ queries, $\mathcal{A}_y$ selects $y^*\|g^*\|h^*$ at random and returns $h^*$ as its new hash in the non-malleability game. Finally, the algorithm $\mathcal{A}_x$ takes $r\|m_b$ as input, selects $m'$ at random, and outputs $r\|m'$.

Now if the event $F_3$ occured, then there is some ciphertext that should not have been rejected, and $\mathcal{A}_3$ will have a $1/q_{\mathcal{D}}$ chance at picking that ciphertext for its output in the NM game. It is simple to check that $F_3$ occurs and $\mathcal{A}_3$ selects this ciphertext, it wins the NM game, giving it a $\varepsilon_3/q_{\mathcal{D}}$ chance at winning, since the first $k/2$ bits coincide. In constrast, any simulator $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_x)$ receives $K' = (K, \mathsf{f})$ as input, chooses the distribution $\mathcal{X}$, and gets only $\mathsf{f}(r)$ as input to $\mathcal{S}_x$. Although this simulator may depend on the relation $R_{\mathsf{BR}}$, it still must choose a canonical encoding of $\mathcal{X}$ for its message distribution, and thus predict the leading $k/2$ bits of $r$. But then it violates partial one-wayness of $\mathsf{f}$. Therefore, the probability of the simulator satisfying the relation $R_{\mathsf{BR}}$ is negligible, and there is noticeable difference to $\mathcal{A}_3$'s success probability.

This yields the contradiction that the hash function is *not* non-malleable for the relation $R_{\mathsf{BR}}$. Thus $\varepsilon_3$ must also be negligible.

COMPARING GAMES THREE AND FOUR Let $S_3$ be the event that $\mathcal{B}$ wins $\mathsf{Game}_3$, and let $S_4$ be the event that it wins $\mathsf{Game}_4$. We claim that $|\Pr[S_3] - \Pr[S_4]| = \varepsilon_4$ is negligible by the POWHF condition in the theorem. To prove this, we construct an adversary $\mathcal{A}_4 = (\mathcal{A}_d, \mathcal{A}_b)$ that wins the POWHF game with probability $\varepsilon_4$. $\mathcal{A}_d$ gets a hash key $K' = (K, \mathsf{f})$ as input, and runs $\mathcal{B}$ with public key $(K, \mathsf{f})$. $\mathcal{A}_d$ answers decryption queries itself, as decribed in $\mathsf{Game}_3$, without $\mathsf{f}^{-1}$. When $\mathcal{B}$ outputs a pair $(m_0, m_1)$, $\mathcal{A}_d$ selects a random bit $b$ and outputs a canonical distribution $\mathcal{X}$ that samples $r \xleftarrow{\$} \{0,1\}^k$ and a random $m'$ from the message space, and outputs $r\|m_b$ with probability $1/2$ and $r\|m'$ otherwise.

$\mathcal{A}_b$ gets as input $\hat{h}$ and $\hat{y} = \mathsf{f}(\hat{r})$. $\mathcal{A}_b$ first chooses a random string $\hat{g}$, which implicitly defines $G(\hat{r}) = \hat{g} \oplus m$, where $m$ is either $m_b$ or a random message chosen by the hash oracle. $\mathcal{A}_b$ continues to simulate the game for $\mathcal{B}$ with challenge ciphertext $\hat{y}, \hat{g}, \hat{h}$ until either (1) $\mathcal{B}$ queries $G(\hat{r})$ or (2) $\mathcal{B}$ halts with a bit $b^*$ as output.

In case (1), $\mathcal{A}_b$ aborts the simulation and can immediately win the POWHF game (with overwhelming probability) by checking if $\mathsf{HVf}(K, \hat{r}\|m_b, \hat{h}) = 1$. In case (2), $\mathcal{A}_b$ tests if $b^* = b$, and if so, it outputs 1, and otherwise it outputs $0$. If the hash oracle returned $\hat{h} = \mathsf{G}(K, r \| m_b)$, then $\mathcal{A}_4$ perfectly simulated $\mathsf{Game}_3$ for the adversary. Otherwise, the adversary perfectly simulated $\mathsf{Game}_4$. A standard argument gives that $\mathcal{A}_4$ has advantage negligibly close to $\varepsilon_4/4$, finishing the proof of the claim.

Finally, it is obvious that $\Pr[\mathcal{B} \text{ wins } \mathsf{Game}_4] = 1/2$ because the bit $b$ is never used, and collecting the relations between all of the games, we get that $\mathcal{B}$ must have had negligible advantage in the original IND-CCA game. ▮

```
**Experiment** $\mathsf{Game}_0(k)$:
$K_{\mathsf{pow}} \xleftarrow{\$} \mathsf{POWK}(1^k)$
$\mathrm{crs} \xleftarrow{\$} \mathsf{CRS}(1^k)$
$K = (K_{\mathsf{pow}}, \mathrm{crs})$
$(\mathcal{X}, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_d(K)$
$x \xleftarrow{\$} \mathcal{X}(1^k)$; $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$
$y_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x; r)$ for $r \xleftarrow{\$} \mathrm{RND}_{K_{\mathsf{pow}}}$
$\pi \xleftarrow{\$} \mathsf{P}(\mathrm{crs}, y_{\mathsf{pow}}, x || r)$
$y = (y_{\mathsf{pow}}, \pi)$
$(y^*, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_y(y, h_x, \mathrm{st}_y)$ where $y^* = (y^*_{\mathsf{pow}}, \pi^*)$
$x^* \xleftarrow{\$} \mathcal{A}_x(x, \mathrm{st}_y)$
Return 1 iff
    $R(\mathcal{X}, x, x^*) = 1 \;\wedge\; (x, y) \neq (x^*, y^*)$
    $\wedge\, \mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$
    $\wedge\, \mathsf{V}(\mathrm{crs}, y^*_{\mathsf{pow}}, \pi^*) = 1$
```

```
**Experiment** $\mathsf{Game}_1(k)$:
$K_{\mathsf{pow}} \xleftarrow{\$} \mathsf{POWK}(1^k)$
$(\mathrm{crs}, \sigma) \xleftarrow{\$} \mathsf{Z}_0(1^k)$
$K = (K_{\mathsf{pow}}, \mathrm{crs})$
$(\mathcal{X}, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_d(K)$
$x \xleftarrow{\$} \mathcal{X}(1^k)$; $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$
$y_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x; r)$ for $r \xleftarrow{\$} \mathrm{RND}_{K_{\mathsf{pow}}}$
$\pi \xleftarrow{\$} \mathsf{Z}_1(\sigma, y_{\mathsf{pow}})$
$y = (y_{\mathsf{pow}}, \pi)$
$(y^*, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_y(y, h_x, \mathrm{st}_y)$ where $y^* = (y^*_{\mathsf{pow}}, \pi^*)$
$x^* \xleftarrow{\$} \mathcal{A}_x(x, \mathrm{st}_y)$
Return 1 iff
    $R(\mathcal{X}, x, x^*) = 1 \wedge (y \neq y^*)$
    $\wedge\, \mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$
    $\wedge\, \mathsf{V}(\mathrm{crs}, y^*_{\mathsf{pow}}, \pi^*) = 1$
```

```
**Experiment** $\mathsf{Game}_2(k)$:
$K_{\mathsf{pow}} \xleftarrow{\$} \mathsf{POWK}(1^k)$
$(\mathrm{crs}, \sigma) \xleftarrow{\$} \mathsf{Z}_0(1^k)$
$K = (K_{\mathsf{pow}}, \mathrm{crs})$
$(\mathcal{X}, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_d(K)$
$x \xleftarrow{\$} \mathcal{X}(1^k)$; $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$
$y_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x; r)$ for $r \xleftarrow{\$} \mathrm{RND}_{K_{\mathsf{pow}}}$
$\pi \xleftarrow{\$} \mathsf{Z}_1(\sigma, y_{\mathsf{pow}})$
$y = (y_{\mathsf{pow}}, \pi)$
$(y^*, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_y(y, h_x, \mathrm{st}_d)$ where $y^* = (y^*_{\mathsf{pow}}, \pi^*)$
$x^* \xleftarrow{\$} \mathcal{A}_x(x, \mathrm{st}_y)$
if $\mathsf{POWVf}(K_{\mathsf{pow}}, x^*, y^*_{\mathsf{pow}}) = 1$
    then $x^*_{\mathsf{K}} || r^*_{\mathsf{K}} \xleftarrow{\$} \mathsf{K}(\sigma, y^*_{\mathsf{pow}}, \pi^*)$
    else $x^*_{\mathsf{K}} \leftarrow \perp$
Return 1 iff
    $R(\mathcal{X}, x, x^*_{\mathsf{K}}) = 1 \;\wedge\; (y \neq y^*)$
    $\wedge\, \mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 1$
    $\wedge\, \mathsf{V}(\mathrm{crs}, y^*_{\mathsf{pow}}, \pi^*) = 1$
```

```
**Experiment** $\mathsf{Game}_3(k)$:
$K_{\mathsf{pow}} \xleftarrow{\$} \mathsf{POWK}(1^k)$
$(\mathrm{crs}, \sigma) \xleftarrow{\$} \mathsf{Z}_0(1^k)$
$K = (K_{\mathsf{pow}}, \mathrm{crs})$
$(\mathcal{X}, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_d(K)$
$x \xleftarrow{\$} \mathcal{X}(1^k)$; $h_x \xleftarrow{\$} \mathsf{hint}(K, x)$
$x' \xleftarrow{\$} \mathcal{X}(1^k)$
$y'_{\mathsf{pow}} \leftarrow \mathsf{POW}(K_{\mathsf{pow}}, x'; r')$ for $r' \xleftarrow{\$} \mathrm{RND}_{K_{\mathsf{pow}}}$
$\pi \xleftarrow{\$} \mathsf{Z}_1(\sigma, y'_{\mathsf{pow}})$
$y = (y'_{\mathsf{pow}}, \pi)$
$(y^*, \mathrm{st}_y) \xleftarrow{\$} \mathcal{A}_y(y, h_x, \mathrm{st}_d)$ where $y^* = (y^*_{\mathsf{pow}}, \pi^*)$
$x^*_{\mathsf{K}} || r^*_{\mathsf{K}} \xleftarrow{\$} \mathsf{K}(\sigma, y^*_{\mathsf{pow}}, \pi^*)$
Return 1 iff
    $R(\mathcal{X}, x, x^*_{\mathsf{K}}) = 1 \;\wedge\; (y \neq y^*)$
    $\wedge\, \mathsf{POWVf}(K_{\mathsf{pow}}, x^*_{\mathsf{K}}, y^*_{\mathsf{pow}}) = 1$
    $\wedge\, \mathsf{V}(\mathrm{crs}, y^*_{\mathsf{pow}}, \pi^*) = 1$
```

Figure 1: Games in the Proof of Theorem 4.2: Shaded areas indicate the differences between the games.