A Random Oracle for All of Us

Marc Fischlin, Felix Rohrbach(✉), and Tobias Schmalz

Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany
{marc.fischlin,felix.rohrbach,tobias.schmalz}@cryptoplexity.de
https://www.cryptoplexity.de

Abstract. We introduce the notion of a universal random oracle. Analogously to a classical random oracle it idealizes hash functions as random functions. However, as opposed to a classical random oracle which is created freshly and independently for each adversary, the universal random oracle should provide security of a cryptographic protocol against *all* adversaries simultaneously. This should even hold if the adversary now depends on the random function. This reflects better the idea that the strong hash functions like SHA-2 and SHA-3 are fixed before the adversary decides upon the attack strategy.

Besides formalizing the notion of the universal random oracle model we show that the model is asymptotically equivalent to Unruh's auxiliary-input random oracle model (Crypto 2007). In Unruh's model the adversary receives some inefficiently computed information about the random oracle as extra input. Noteworthy, while security in the universal random oracle model implies security in the auxiliary-input random oracle model tightly, the converse implication introduces an inevitable security loss. This implies that the universal random oracle model provides stronger guarantees in terms of concrete security. Validating the model we finally show, via a direct proof with concrete security, that a universal random oracle is one-way.

1 Introduction

The random oracle methodology [2,11] has turned out to be a useful tool to design cryptographic protocols with practical efficiency, while allowing security proofs if one assumes that the hash function behaves ideally. That is, in the security proof one assumes that the involved hash function acts optimally like a random function. The underlying assumption is that, if one later uses a strong hash function like SHA-2 or SHA-3 in practice, then any attack against the protocol must be due to unexpected weakness in the hash function. While the soundness of this approach has been disputed, e.g., [1,7,14,17], we have not yet experienced practical schemes showing such weaknesses (i.e., without incorporating an obvious structural shortcoming in this regard).

1.1 The Universal Random Oracle Model

Security in the random oracle model considers executions where the random oracle is chosen when the attack starts, independently of the adversary and its

L. Batina and J. Daemen (Eds.): AFRICACRYPT 2022, LNCS 13503, pp. 469–489, 2022.
https://doi.org/10.1007/978-3-031-17433-9_20

strategy. If one goes back to the original idea of later plugging in SHA-2 or SHA-3, however, the order compared to practical settings is in fact reversed: These hash functions have been designed first and are already available, such that the adversary may actually take advantage of this a-priori knowledge in the attack. In contrast, common security games first fix the adversary and then initialize the random oracle.

At first it seems as if the idea of making the adversary depend on the random oracle would refute the idea of eliminating the presence of any structural weakness of the hash function. But recall that we still consider the random oracle to be a random function, only that the adversary may now depend on this random function. In a sense, the adversary still cannot exploit functional properties of the hash functions, but it may take into account that the actual hash function in the protocol is fixed before the protocol is attacked, or even designed. We call this a *universal* random oracle, because the same random oracle should work against all adversaries.

On a technical level the difference between the two approaches, the classical random oracle model and the universal one, becomes apparent through the usage of the Borel-Cantelli lemma, as done for example in the famous work by Impagliazzo and Rudich [16]. The Borel-Cantelli lemma allows to reverse the order of quantifiers in the sense that if for any adversary the probability of an attack for a random oracle is negligible, then there exists an oracle which works against all adversaries. In fact, a random oracle will work almost surely against all adversaries.

Hence, while one could use in principle the Borel-Cantelli lemma to switch from the random oracle model to the universal one, the lemma comes with two disadvantages. First, the final step in the argument, namely that a random function works against all adversaries, only works against the countable class of uniform adversaries (unless one makes further restrictions on the security reduction itself [3,4]), and thus excludes non-uniform adversaries. The second disadvantage is that the asymptotic statement of the Borel-Cantelli lemma infringes with the notion of concrete security. But the latter is important for schemes aiming at practicality. Hence, using the Borell-Cantelli lemma in principle indeed allows to go from classical random oracles to universal ones, but comes at a price.

1.2 Defining Universal Random Oracles

Defining the universal random oracle model (UROM) is more challenging than one would envision. A straightforward approach would be to demand that for a random oracle $\mathcal{O}$ no adversary A can win the corresponding security experiment Game (with more than negligible probability ε in the security parameter λ):

$$\mathbb{P}_{\mathcal{O}}\left[\forall A\ \exists \varepsilon \in \mathrm{negl}\ \forall \lambda :\ \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon(\lambda)\right] = 1.$$

However, as we argue this definition appears to be too liberal: We provide an experiment which was secure in this version of the UROM, although the experiment is both intuitively insecure and also provably breakable in the ordinary

random oracle model. This would violate our intuition that the UROM provides stronger security guarantees than the ordinary ROM.

The above mismatch also motivates our actual definition of the UROM. As in the random oracle model we aim for security for a given security parameter, such that the quantification over λ appears outside of the probability over the random oracle:

$$\forall s \in \text{poly}\ \exists \varepsilon \in \text{negl}\ \forall \lambda : \\ \mathbb{P}_{\mathcal{O}}\left[\forall A \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A,\mathcal{O}}(\lambda)\right] \leq \varepsilon(\lambda)\right] \geq 1 - \varepsilon(\lambda).$$

To let the adversary A depend on the random oracle we quantify over all adversaries in the probability for $\mathcal{O}$, and only use a size bound $s(\lambda)$ on the outside. We give more details about this choice within. Another justification for the correctness of our approach is by relating this model to existing definitions, especially to the auxiliary-input random oracle model.

1.3 Relationship to Auxiliary-Input Random Oracles

Unruh [19] defined the auxiliary-input random oracle model (AI-ROM) as an extension of the classical random oracle model. In this model the adversary A receives as input some information about the previously sampled random oracle $\mathcal{O}$, provided by some unbounded algorithm $z^{\mathcal{O}}$ with oracle access to the random oracle. This can, for example, be a collision found in exponential time such that random oracles are not collision-resistant in this model.

Unruh's main technical result, called lazy sampling with auxiliary input [19, Theorem 2], is to relate the statistical distance of outputs for adversaries receiving auxiliary input $z^{\mathcal{O}}$ for random oracle $\mathcal{O}$, to the one when instead having access to a fresh random oracle (but which is consistent on some fixed values with the original oracle). He shows that the statistical distance between the two settings is of order $\mathcal{O}\left(\sqrt{ST/P}\right)$ where S is the bit size of auxiliary information, T is the number or random oracle queries of the adversary, and P is the number of coordinates to be fixed. The bound was subsequently improved to $\mathcal{O}\left(ST/P\right)$ by Coretti et al. [9], matching a lower bound of Dodis et al. [10].

Here we show that the two models, AI-ROM and UROM, are equivalent. That is, if a game is secure in one model, then it is also secure in the other model. Remarkably, there is a security degradation when going from AI-ROM to UROM: If a game is ϵ-secure in the AI-ROM, then it is "only" $\sqrt{\epsilon}$-secure in the UROM. We also show that this quadratic loss is inevitable in general. The other direction holds tightly, i.e., ϵ-security in the UROM implies ϵ-security in the AI-ROM. In this sense, the UROM gives stronger security guarantees for concrete bounds.

Another interesting aspect of UROM is that the separation of the game's randomness from the randomness of choosing the random oracle allows for more freedom in setting the security bounds. The above equivalence of UROM and AI-ROM holds for negligible bounds for both the game's and the random oracle's

randomness, but the UROM model also allows for notions where the game should have a negligible success probability for any adversary, with all but exponentially-small probability in the selection of the random oracle.

1.4 Relationship to Global Random Oracles

Canetti et al. [8], and later Camenisch et al. [5], considered the notion of global random oracles in the Universal Composability (UC) framework [6]. The starting point of the global random oracle model is the observation that, even if multiple components of a cryptographic protocols are proven UC secure in the (standard) random oracle model, their composition is not necessarily secure if the random oracle is replaced by the same hash function in all components. The global random oracle model now says that a single, global random oracle functionality is available to all parties. A security proof in the model allows for one random oracle to be used in all components, and therefore also to be replaced with the same hash function everywhere.

The global random oracle model and the UROM are close in spirit in light of the idea that the same hash function may be used at several places, but are technically somewhat orthogonal. The global random oracle model is investigating cross-effects of hash function deployments between different protocol executions (but a fresh random oracle instance is chosen when considering an attack on the composed setting). In contrast, the UROM is concerned with dependencies of the adversary with respect to the universally available random oracle within an abstract security game. This abstract security game may be compound of several protocols but is not cast in a simulation-based setting like the UC framework.

1.5 Proving Security in the UROM

We finally give an example of how to show security in the UROM, by showing that one-wayness exists in the UROM. Of course, we can immediately transfer any security result from the AI-ROM via the equivalence, however, this example lets us use the above mentioned flexibility in choosing our security bounds to show that one-wayness exists for all but exponentially few (universal) random oracles.

The proof of one-wayness follows the compression technique of Gennaro and Trevisan [12]. Similar approaches have also been given for the AI-ROM [9,10]; our goal here is to exercise security arguments in the UROM model. The line of reasoning is as follows. If there was a successful adversary against the one-wayness of the UROM, then we can compress the random oracle with the help of the adversary. The important point here is that the adversary may depend on the random oracle for this compression, making the approach employable in the UROM. If the adversary is too successful then we can actually compress beyond information-theoretic lower bounds. Of course, we state the latter fact in terms of concrete security in the UROM.

2 Preliminaries

In this section we present the basic notions of negligible functions, security games, and the (classical) random oracle model, and one-wayness. The notions of the UROM and AI-ROM are given in the subsequent sections.

2.1 Negligible Functions

We use the standard of notion of negligible function and state some very basic but useful properties afterwards:

Definition 1 (Negligible Functions). *A function* $\varepsilon : \mathbb{N} \to \mathbb{R}$ *is called* negligible *if for any polynomial* $p : \mathbb{N} \to \mathbb{R}^+$ *there exists* $\Lambda \in \mathbb{N}$ *such that*

$$\forall \lambda \geq \Lambda : \varepsilon(\lambda) \leq \frac{1}{p(\lambda)}.$$

We denote the set of all negligible functions by negl *and the set of all functions which are not negligible by* non-negl.

Note that we allow negligible functions ε to be negative at some inputs. When quantifying over all negligible functions we can restrict ourselves to non-negative functions by considering the pointwise maximum $\max\{0, \varepsilon(\lambda)\}$. If and only if ε is negligible so is this maximum. Analogously, we can always presume $\varepsilon(\lambda) \leq 1$ when quantifying over all negligible functions. This follows by considering the pointwise minimum $\min\{1, \varepsilon(\lambda)\}$.

When considering definitions we sometimes bound success probabilities for a sequence of events E_λ (e.g., an adversary winning a security game for parameter λ), by negligible functions:

$$\exists \varepsilon \in \text{negl}\, \forall \lambda \in \mathbb{N} :\ \mathbb{P}[E_\lambda] \leq \varepsilon(\lambda).$$

Note that we can quantify over all λ since we can change the negligible function ε at finitely many points. When negating this statement, e.g., when describing that there exists a successful adversary, we get

$$\forall \varepsilon \in \text{negl}\, \exists \lambda \in \mathbb{N} :\ \mathbb{P}[E_\lambda] > \varepsilon(\lambda). \tag{1}$$

This means that for any (negligible) bound we find a security parameter where the probability of the event, e.g., the adversary winning, exceeds this bound. When showing our relationship of the UROM to the AI-ROM we use that the above holds if and only if

$$\forall \varepsilon \in \text{negl}\, \exists \lambda \in \mathbb{N} :\ \mathbb{P}[E_\lambda] > \varepsilon^2(\lambda). \tag{2}$$

To see this note that we may only consider non-negative functions ε bounded from above by 1. But then the function $\delta(\lambda) := \sqrt{\varepsilon(\lambda)}$ is well defined and it holds that $\delta(\lambda) \geq \varepsilon(\lambda)$ for all security parameters. Furthermore, δ is negligible if and only if ε is.

Hence, when quantifying over all negligible functions we can always switch between ε and δ and get the desired bound. More precisely, assume that statement (1) holds. Take some arbitrary negligible function ε. Our goal is to show that there exists some λ such that $\mathbb{P}[E_\lambda] > \varepsilon^2(\lambda)$. But this follows straightforwardly from the first statement since $\varepsilon(\lambda) \geq \varepsilon^2(\lambda)$. For the converse note that, if statement (2) holds, then for any given negligible function ε we can consider the function $\delta(\lambda) = \sqrt{\varepsilon(\lambda)}$ in the second statement. By assumption there exists some λ where the probability exceeds $\delta^2(\lambda) = \varepsilon(\lambda)$. This shows that the first statement holds in this case as well for any negligible function.

2.2 Security Games

We consider abstract security games involving the adversary A and an oracle $\mathcal{O}$. We denote by $\mathsf{Game}^{A,\mathcal{O}}(\lambda)$ the binary outcome of executing the security game. We often emphasize the dependency of algorithms and functions by using subscripts, e.g., we write $A_\mathcal{O}$ to denote the fact that the adversary may depend on oracle $\mathcal{O}$, or $\varepsilon_{A,\mathcal{O}}(\lambda)$ to indicate that the function ε depends on both the adversary and the oracle. We use the terms security game and experiment interchangeably.

Further, by $A^\mathcal{O}$, we denote that the adversary A has oracle-access to $\mathcal{O}$. We view A as a (family of) circuit(s) that have a special oracle gate, which allows A to query the oracle. We capture adversaries with an upper bound $s(\lambda)$ of the size for non-uniform adversaries resp. run time for uniform adversaries via a set $\mathrm{SIZE}(s(\lambda))$. The set of efficient adversaries is given by $\mathrm{SIZE}(\mathrm{poly})$.

We write $\mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^\mathcal{O},\mathcal{O}}(\lambda)\right]$ for the probability that adversary A with access to random oracle $\mathcal{O}$ wins in the security game Game. Here, the probability is over all random choices in the game, including the randomness of the adversary. The random oracle, however, is fixed at this point and the adversary may depend on $\mathcal{O}$. The oracle is usually chosen "outside" of the game.

2.3 The Random Oracle Model

A random oracle $\mathcal{O}$ is an oracle that gives access to a truly random function. We assume that for every security parameter λ, oracle $\mathcal{O}$ maps inputs from $\{0,1\}^*$ or from $\{0,1\}^{\leq d(\lambda)}$ to outputs from $\{0,1\}^\lambda$, where $\{0,1\}^{\leq d(\lambda)}$ denotes the set of strings of bit length at most $d(\lambda)$. For so-called length-preserving random oracles the domain for every security parameter λ is simply $\{0,1\}^\lambda$, i.e., inputs are of length $d(\lambda) = \lambda$ exactly. In the classical random oracle model we pick the oracle $\mathcal{O}$ as part of the game. In this case we usually write $\mathbb{P}_{\mathsf{Game},\mathcal{O}}\left[\mathsf{Game}^{A^\mathcal{O},\mathcal{O}}(\lambda)\right]$ for the probability of A winning the corresponding game. Note that here now A usually does not depend on $\mathcal{O}$ beyond the oracle access.

With the above notation we can phrase the (classical) random oracle model as follows.

Definition 2 (ROM). *An experiment* Game *is secure in the ROM iff*

$$\forall A \in \mathrm{SIZE}(\mathrm{poly})\ \exists \varepsilon \in \mathrm{negl}\ \forall \lambda \in \mathbb{N}:\ \mathbb{P}_{\mathcal{O},\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon(\lambda).$$

2.4 One-Wayness in the Random Oracle Model

To define that a random oracle $\mathcal{O}$ immediately gives a one-way function we simply state the security game $\mathsf{Game}_{\mathsf{OW}}^{A^{\mathcal{O}},\mathcal{O}}(\lambda)$ as follows. Run $A^{\mathcal{O}}(1^\lambda, \mathcal{O}(x))$ for $x \leftarrow\!\!\$\ \{0,1\}^\lambda$ to obtain a value x^*. Let the game output 1 if and only if $\mathcal{O}(x) = \mathcal{O}(x^*)$. If we assume length-preserving random oracles, as in Sect. 5, we necessarily have $x^* \in \{0,1\}^\lambda$ then:

$\mathsf{Game}_{\mathsf{OW}}^{A,\mathcal{O}}(\lambda)$

1: $x \leftarrow\!\!\$\ \{0,1\}^\lambda$
2: $x^* \leftarrow\!\!\$\ A^{\mathcal{O}}(1^\lambda, \mathcal{O}(x))$
3: **return** 1 **if** $\mathcal{O}(x^*) = \mathcal{O}(x)$ **else** 0

Note that if we later switch to the UROM, then the one-wayness security game does not change, only the oracle setting.

3 The Universal Random Oracle Model UROM

A straightforward formalization of the universal ROM may now be to demand that, with probability 1, a random oracle $\mathcal{O}$ is good for all adversaries A. That is, for each adversary the success probability is negligible for the given random oracle:

$$\mathbb{P}_{\mathcal{O}}\left[\begin{matrix}\forall A_{\mathcal{O}} \in \mathrm{SIZE}(\mathrm{poly}(\lambda)) \\ \exists \varepsilon_{A,\mathcal{O}} \in \mathrm{negl}\ \forall \lambda \in \mathbb{N}\end{matrix} : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_{A,\mathcal{O}}(\lambda)\right] = 1.$$

The issue with this approach is that it identifies a case which is both intuitively insecure and also provably insecure in the (plain) random oracle, to be secure in this version of the UROM, as we discuss in Appendix A.

We next present the, in our view, right definition of the UROM. The essence from the above failed definitional attempt is that we have to pull out the quantification of the security parameter from the outer probability, and therefore all preceding quantifiers. But moving out the quantification over all adversaries would infringe with our idea to make the adversary depend on the random oracle. To re-install this idea we set a bound on the adversarial success probability and run time resp. size, and define "good" random oracles for all adversaries within such bounds:

Definition 3 (UROM). *A security game* Game *is secure in the UROM if*

$$\forall s \in \mathrm{poly}\ \exists \varepsilon_s \in \mathrm{negl}\ \forall \lambda \in \mathbb{N}:$$
$$\mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \mathrm{SIZE}(s(\lambda)) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_s(\lambda)\right] \geq 1 - \varepsilon_s(\lambda). \quad (3)$$

Note that with the "outer" negligible error function ε_s we account for "bad" random oracles, for which the security game may be easy to win, e.g., breaking one-wayness for the all-zero oracle $\mathcal{O}$. Since we may consider finite domains and ranges for $\mathcal{O}$ for the fixed-size adversaries (for given security parameter λ), such bad random oracles may have a non-zero probability. The negligible function expresses that such oracles are very sparse. As the weakness of the random oracle may depend on the adversarial size, e.g., many hardcoded preimages in the one-wayness experiment may affect the security, we make the negligible function also depend on the size s. The "inner" probability then captures that no adversary (of the given size) can win the security game with high probability, but here the probability is only over all the random choices of the game and adversary.

Both the inner and outer negligible error terms are based on the same function. We could have chosen different negligible functions ε_s (for the inner game probability) and δ_s for the outer oracle probability, and quantify over the existence of such negligible functions ($\exists \varepsilon_s, \delta_s \in \text{negl}$). But the pointwise maximum, $\gamma_s(\lambda) := \max\{\varepsilon_s(\lambda), \delta_s(\lambda)\}$, would also be negligible and satisfy the bounds:

- For $\gamma_s(\lambda) = \delta_s(\lambda) \geq \varepsilon_s(\lambda)$ we have

$$\begin{aligned} 1 - \delta_s(\lambda) &\leq \mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_s(\lambda)\right] \\ &\leq \mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \delta_s(\lambda)\right]. \end{aligned}$$

- For $\gamma_s(\lambda) = \varepsilon_s(\lambda) \geq \delta_s(\lambda)$ we have

$$\begin{aligned} \mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_s(\lambda)\right] &\geq 1 - \delta_s(\lambda) \\ &\geq 1 - \varepsilon_s(\lambda). \end{aligned}$$

It thus suffices to consider a single negligible function.

4 UROM vs. AI-ROM

In this section we show that UROM and AI-ROM are equivalent, although not tightly related. We first present the AI-ROM and then show both directions of the equivalence.

4.1 AI-ROM

The auxiliary-input random oracle model AI-ROM [19] allows a preprocessing through an unbounded oracle algorithm $z^{(\cdot)}$ which on input the security parameter outputs a polynomial-size string. This string is then given as auxiliary information about the random oracle to the adversary A. Experiments in which the adversary needs to find a collision in the (unkeyed) random oracle are for example insecure in the AI-ROM: The function $z^{\mathcal{O}}(1^\lambda)$ exhaustively searches for a collision $x \neq x'$ of complexity λ and outputs this pair; adversary A simply outputs the collision. This is in contrast to security in the regular ROM in which no efficient A is able to find such a collision with non-negligible probability.

Definition 4 (AI-ROM). *An experiment Game is secure in the AI-ROM iff*

$$\forall A, z^{(\cdot)}\ \exists \varepsilon \in \text{negl}\ \forall \lambda : \mathbb{P}_{\mathcal{O},\mathit{Game}}\left[\mathit{Game}^{A^{\mathcal{O}}(z^{\mathcal{O}}),\mathcal{O}}(\lambda)\right] \leq \varepsilon(\lambda).$$

We will now show the equivalence of the two notions. Section 4.2 will show that AI-ROM implies UROM, and Sect. 4.3 will show the reverse implication.

4.2 AI-ROM Implies UROM

Theorem 1 (AI-ROM ⇒ UROM). *If Game is secure in the AI-ROM then it is also secure in the UROM.*

Proof. Assume an experiment Game is insecure in the UROM, i.e., negating the security requirement we have

$$\exists s \in \text{poly}\ \forall \varepsilon \in \text{negl}\ \exists \lambda \in \mathbb{N} :$$
$$\mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon(\lambda)\right] < 1 - \varepsilon(\lambda).$$

We will show that the experiment is also insecure in the AI-ROM, by constructing an adversary pair (A_{AI}, z) against the auxiliary-input setting. First, to get a better intuition we switch to the complementary event of the outer probability for our successful attack again the UROM:

$$\exists s \in \text{poly}\ \forall \varepsilon \in \text{negl}\ \exists \lambda \in \mathbb{N} :$$
$$\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] > \varepsilon(\lambda)\right] \geq \varepsilon(\lambda).$$

This formula now states the fraction of "bad" random oracles $\mathcal{O}$ for which there exists a successful adversary, exceeding the bound $\varepsilon(\lambda)$ for some parameter λ, cannot be upper bounded by any negligible function $\varepsilon(\lambda)$. We can capture this set $\Omega = \Omega_{s,\varepsilon,\lambda}$ of bad random oracles as

$$\Omega \coloneqq \left\{\mathcal{O} \;\middle|\; \exists A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] > \varepsilon(\lambda)\right\}$$

Note that all parameters s, ε, λ are fixed via the quantifiers when defining this set. This is especially true for the security parameter λ, so the condition is *not* a statement over all security parameters. To emphasize this we can also write the definition of Ω implicitly as

$$\exists s \in \text{poly}\ \forall \varepsilon \in \text{negl}\ \exists \lambda \in \mathbb{N}\ \forall \mathcal{O} \in \Omega\ \exists A_{\mathcal{O}} \in \text{SIZE}\left(s(\lambda)\right) :$$
$$\mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] > \varepsilon(\lambda).$$

Note that for the fixed values s, ε, λ we have $\mathbb{P}_{\mathcal{O}}\left[\mathcal{O} \in \Omega\right] \geq \varepsilon(\lambda)$ and therefore

$$\mathbb{P}_{\mathcal{O},\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \geq \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathcal{O}},\mathcal{O}}(\lambda) \mid \mathcal{O} \in \Omega\right] \cdot \mathbb{P}_{\mathcal{O}}\left[\mathcal{O} \in \Omega\right] > \varepsilon^2(\lambda).$$

Next we define an inefficient function $z_s^{(\cdot)}$ that, given any oracle $\mathcal{O}$ and security parameter λ, outputs (a circuit description of) the adversary $A_{\mathcal{O}}$ of size at most $s(\lambda)$ with the highest success probability against the game. This adversary will win the game with probability more than $\varepsilon(\lambda)$ for any oracle in Ω according to the definition. Further, we define A_{AI}, which is the universal circuit that will interpret the circuit returned by z_s. Note that the universal circuit A_{AI} has polynomial size, since the size of $A_{\mathcal{O}}$ is bounded by the (fixed) polynomial $s(\lambda)$ and the execution of a circuit can be done efficiently. Therefore we can conclude that there exists a pair (A_{AI}, z_s), where A_{AI} is polynomially bounded. This pair is successful in the given game for any bad oracle from the set Ω:

$$\exists(A_{\mathsf{AI}}, z_s^{(\cdot)})\ \forall\varepsilon \in \mathrm{negl}\ \exists\lambda \in \mathbb{N}\ \forall\mathcal{O} \in \Omega : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathsf{AI}}^{\mathcal{O}}(z_s^{\mathcal{O}}),\mathcal{O}}(\lambda)\right] > \varepsilon(\lambda).$$

It remains to show that a similar bound also holds when we go back to picking $\mathcal{O}$ at random from *all* random oracles instead of the set Ω. To this end we use our assumption that the probability of an oracle being in Ω is at least $\varepsilon(\lambda)$:

$$\exists(A_{\mathsf{AI}}, z_s^{(\cdot)})\ \forall\varepsilon \in \mathrm{negl}\ \exists\lambda \in \mathbb{N} : \mathbb{P}_{\mathcal{O},\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathsf{AI}}^{\mathcal{O}}(z_s^{\mathcal{O}}),\mathcal{O}}(\lambda)\right] > \varepsilon^2(\lambda).$$

According to the discussion about negligible functions we can rewrite this as

$$\exists(A_{\mathsf{AI}}, z_s^{(\cdot)})\ \forall\varepsilon \in \mathrm{negl}\ \exists\lambda \in \mathbb{N} : \mathbb{P}_{\mathcal{O},\mathsf{Game}}\left[\mathsf{Game}^{A_{\mathsf{AI}}^{\mathcal{O}}(z_s^{\mathcal{O}}),\mathcal{O}}(\lambda)\right] > \varepsilon(\lambda).$$

But this means that the protocol cannot be secure in the AI-ROM. □

In terms of exact security, the derived adversary A_{AI} has roughly the same running time as A. But its success probability drops from $\varepsilon(\lambda)$ (for A) to $\varepsilon^2(\lambda)$. In general this is inevitable, though. For any $k = \omega(\lambda)$ consider the game $\mathsf{Game}^{A,\mathcal{O}}(\lambda)$ which returns 1 if the leading k bits of $\mathcal{O}(0^\lambda)$ are all 0 and if, in addition, k random coin flips also land all on 0. Then the probability that any pair (A_{AI}, z) wins in the AI-ROM setting is at most 2^{-2k}. But in the UROM setting we can set $\varepsilon(\lambda)$ to be 2^{-k}, because for all "good" oracles $\mathcal{O}$ with the leading k bits of $\mathcal{O}(0^\lambda)$ being different from 0 no adversary can win the game.

4.3 UROM Implies AI-ROM

Theorem 2 (UROM ⇒ AI-ROM, tightly). *If Game is secure in the UROM then it is also secure in the AI-ROM.*

For the proof we use the so-called splitting lemma [18] which allows to relate the probability of events over a product space $X \times Y$ to the ones when the X-part is fixed:

Lemma 1 (Splitting Lemma [18]). *Let $\mathcal{D} = \mathcal{D}_X \times \mathcal{D}_Y$ be some product distributions over $X \times Y$. Let $Z \subseteq X \times Y$ be such that $\mathbb{P}_{\mathcal{D}}[(x,y) \in Z] > \varepsilon$. For any $\alpha < \varepsilon$ call $x \in X$ to be α-good if*

$$\mathbb{P}_{y \leftarrow\$ \mathcal{D}_Y}[(x,y) \in Z] > \varepsilon - \alpha.$$

Then we have $\mathbb{P}_{x \leftarrow\$ \mathcal{D}_X}[x \text{ is } \alpha\text{-good}] \geq \alpha$.

Proof (of Theorem 2). Assume that we have a successful attacker in the AI-ROM:

$$\exists(A_{\mathsf{AI}}, z^{(\cdot)})\ \forall\varepsilon\in\mathrm{negl}\ \exists\lambda\in\mathbb{N}:\mathbb{P}_{\mathcal{O},\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathsf{AI}}(z^{\mathcal{O}}),\mathcal{O}}(\lambda)\right]>\varepsilon(\lambda).$$

We show that we can build a successful adversary A in the UROM model. We first apply the splitting lemma (Lemma 1) for fixed $A_{\mathsf{AI}}, z, \varepsilon, \lambda$. We will only consider such choices which exceed the bound $\varepsilon(\lambda)$. We define the distribution $\mathcal{D}_X$ as the choice of a random oracle $\mathcal{O}$, and $\mathcal{D}_Y$ as the randomness in the game (for both the game and the adversary), as well as Z as the events in which (A_{AI}, z) wins the game for the random oracle. This happens with probability at least $\varepsilon(\lambda)$ by assumption. If we now choose α to be $\frac{1}{2}\varepsilon$ then we get that $\mathbb{P}_{\mathcal{O}}\left[\mathcal{O}\text{ is }\alpha\text{-good}\right]\geq\frac{1}{2}\varepsilon$. Therefore,

$$\exists(A_{\mathsf{AI}}, z)\ \forall\varepsilon\in\mathrm{negl}\ \exists\lambda\in\mathbb{N}:\ \mathbb{P}_{\mathcal{O}}\left[\mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathsf{AI}}(z),\mathcal{O}}(\lambda)\right]>\tfrac{1}{2}\varepsilon\right]\geq\tfrac{1}{2}\varepsilon$$

As $\frac{1}{2}\varepsilon$ is negligible iff ε is, and since we quantify over all negligible functions, we get

$$\exists(A_{\mathsf{AI}}, z)\ \forall\varepsilon\in\mathrm{negl}\ \exists\lambda\in\mathbb{N}:\ \mathbb{P}_{\mathcal{O}}\left[\mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathsf{AI}}(z^{\mathcal{O}}),\mathcal{O}}(\lambda)\right]>\varepsilon(\lambda)\right]\geq\varepsilon(\lambda).$$

Since A_{AI} is polynomially bounded, and $z^{(\cdot)}$ only returns a polynomial-size string, we can view A_{AI} as a circuit of polynomial size $s(\lambda)$. But then we can interpret the AI-ROM adversary pair as consisting of an oracle-dependent component, namely the polynomial-size string $z^{\mathcal{O}}$, and a general part A_{AI}. If we hardcode the string $z^{\mathcal{O}}$ we can write this as a single oracle-dependent adversary $A_{\mathcal{O}}$ of polynomial size $s(\lambda)$. Moving this oracle-dependent algorithm inside the outer probability we obtain:

$$\exists s(\lambda)\in\mathrm{poly}\ \forall\varepsilon\in\mathrm{negl}\ \exists\lambda\in\mathbb{N}:$$
$$\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}}\in\mathrm{SIZE}\left(s(\lambda)\right):\mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathcal{O}},\mathcal{O}}(\lambda)\right]>\varepsilon(\lambda)\right]\geq\varepsilon(\lambda).$$

This shows that the have a successful adversary against the UROM. □

Remarkably, the reduction here is tight. If we have an adversary A_{AI} and z against the AI-ROM, then we get a successful adversary A against UROM with the same running time (as AI-ROM) and, except for a factor $\frac{1}{2}$, the same success probability. This shows that the AI-ROM and UROM model are qualitatively equivalent. Yet, quantitatively, a security bound in the AI-ROM may be significantly looser than in the UROM (see the discussion after Theorem). This means that a direct proof in the UROM may yield tighter bounds.

4.4 Advantages of UROM

While AI-ROM and UROM are equivalent as shown in the last two sections, we argue that UROM has some advantages over AI-ROM, as it provides more flexibility in choosing security bounds. By having separate bounds for the selection

of the random oracle and the success probability of an adversary in the security game, we can for example demand that a game might only be won with negligible probability, for all but an exponential fraction of random oracles. Or, conversely, we could show that a game is secure for every second random oracle, if we can be reasonably sure that we can use one of the good oracles, while in the AI-ROM, a proof might not be possible at all.

For the former, we will give an example in the next section, showing that UROM is a one-way function for nearly all oracles.

5 Universal Random Oracles are One-Way Functions

In this chapter, we will show that random oracles exist in the UROM (more specifically, that the oracle itself is a one-way function). This result serves as an example how to prove security in the UROM, and how to show that a game is secure for all but an exponential fraction of random oracles. Our proof will use the compression technique introduced by Gennaro and Trevisan [12], although our notation is closer to the argument by Haitner et al. [15].

We note that similar results exist for the AI-ROM [9,10], which shows that in the AI-ROM, a one-way function exists which no adversary can invert with probability higher than $\frac{AT}{2^\lambda} + \frac{T}{2^\lambda}$, where A denotes the size of the non-uniform advice the adversary gets about the oracle, and T denotes the number of queries to the oracle. Obviously, their result could be translated to a security bound in the UROM due to the equivalence of the two notions, but the goal here is to present a proof that directly works in the UROM.

The idea of the proof is that, if we have a successful adversary against the random oracle $\mathcal{O}$, then we can use this (specific) adversary to compress the oracle $\mathcal{O}$ into a smaller description, contradicting lower bounds for the description size of random oracles. The reason that this works in the UROM is that the compression can of course depend on the random oracle, such that the adversary, too, can depend on $\mathcal{O}$.

For simplicity reasons, we will assume for this chapter that the random oracle $\mathcal{O}$ is always length-preserving (i.e., $d(\lambda) = \lambda$). Note that the existence of length-preserving one-way functions is equivalent to the existence of general one-way functions [13], so this assumption does not influence the result.

We state the result in terms of exact security, using the general UROM approach where we have different probabilities for the inner and outer probability (for the game hardness resp. for the random oracle):

Theorem 3. *Let S be the maximum size of an adversary. Then, a random oracle model is a one-way function UROM with security bounds $\frac{1}{P}$ and $2^{-\lambda}$ for the inner and outer probability, under the condition that $P \cdot S \leq 2^{\lambda/4}$ and $\lambda > 55$:*

$$\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}} \in \mathrm{SIZE}\,(S) : \mathbb{P}_{x \leftarrow \{0,1\}^\lambda}\left[A^{\mathcal{O}}_{\mathcal{O}}(1^\lambda, \mathcal{O}(x)) \in \mathcal{O}^{-1}(\mathcal{O}(x))\right] > \frac{1}{P}\right] < 2^{-\lambda}$$

The asymptotic version follows as an easy corollary:

Corollary 1. *UROM is a one-way function in the* UROM *model: For every polynomial $s(\lambda)$ bounding the size of an adversary, there exists a negligible function $\epsilon_s(\lambda)$ such that for all security parameters λ,*

$$\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}} \in \mathrm{SIZE}\left(s(\lambda)\right) : \mathbb{P}_{x \leftarrow \{0,1\}^\lambda}\left[A_{\mathcal{O}}^{\mathcal{O}}(1^\lambda, \mathcal{O}(x)) \in \mathcal{O}^{-1}(\mathcal{O}(x))\right] > \varepsilon_s(\lambda)\right] < 2^{-\lambda}$$

Our compression argument will work as follows: Assuming that the random oracle $\mathcal{O}$ in the UROM is not a one-way function, we will show that we can describe the oracle $\mathcal{O}$ with less bits than should be required for a truly random function. For this we assume that A is deterministic; if it is not, then we can make it deterministic by hard-coding the best randomness. We also assume that A needs to output a preimage of size λ and thus only makes queries of this size; any other queries do not help to find a preimage of λ bits and could be easily answered randomly by A itself. Both of these assumptions only increase the size of A at most by a small, constant factor which does not affect our proof.

We give an encoder algorithm which encodes the entire UROM-oracle $\mathcal{O}$ using the successful adversary A, as well as a decoder algorithm which reconstructs $\mathcal{O}$ without access to the oracle itself, using the shorter output of the encoder only. The code for both algorithms is given in Fig. 1. The encoder starts by defining the set I of all images y on which $A_{\mathcal{O}}$ is able to find some preimage x. Note that, as $A_{\mathcal{O}}$ is deterministic, for given $\mathcal{O}$, we can indeed specify if $A_{\mathcal{O}}$ is successful on some input y or not. Further, the encoder creates two initially empty sets: Y, which will contain all the y's for which we reply on $A_{\mathcal{O}}$ to recover one of y's preimages (and which we therefore do not have to save explicitly); and Z, which will contain all full pairs (x, y) with $\mathcal{O}(x) = y$. Therefore, Y denotes the set for which values we actually compress (by not saving the corresponding x-values).

As long as the set I of invertible images still contains values, the encoder takes the lexicographically smallest value y out of I and adds it to Y. We simply write $\min I$ for this element (line 4). Now, the encoder emulates a run of $A_{\mathcal{O}}$ with y as input and checks for all queries. There are two types of queries we need to take care of: The first one are hitting queries, i.e., queries to $\mathcal{O}$ which return y (line 8). In this case, however, A has already given us the preimage, therefore, we abort the simulation at this point. The second type of queries we have to handle are queries that return values y which are still in I (line 10). To make sure we have no circular dependencies between these values, we remove these values from the set I. After the execution of $A_{\mathcal{O}}$ finishes and found a preimage x, we add all further preimages $x' \neq x$ of y that $A_{\mathcal{O}}$ did not return to Z and continue (line 14). Finally, after the set I has become empty, we add all preimages of $y \notin Y$ (as pairs with the image) to Z (line 16). The encoder eventually returns the sets Y, Z, plus a description of $A_{\mathcal{O}}$.

The decoder, on input Y, Z and $A_{\mathcal{O}}$, starts by initializing $\mathcal{O}$ with all the preimage-image-pairs in Z (line 1). Now, similar to the encoder, the decoder goes through all values in Y in lexicographical order and emulates a run of $A_{\mathcal{O}}$ using the partial definition of $\mathcal{O}$. Note that at this point, we already have a partial description of $\mathcal{O}$ that consists of all value-image-pairs we got via Z as well as all preimages we reconstructed in previous steps. Therefore, for each

$\text{Encoder}^{\mathcal{O}}(1^\lambda, A_{\mathcal{O}})$

1 : $I \leftarrow \{y \in \{0,1\}^\lambda | A^{\mathcal{O}}_{\mathcal{O}}(y) \text{ successful}\}$
2 : $Y, Z \leftarrow \emptyset$
3 : **while** $I \neq \emptyset$:
4 : $\quad y \leftarrow \min I$
5 : $\quad Y \leftarrow Y \cup \{y\},\ I \leftarrow I \setminus \{y\}$
6 : $\quad$ Emulate $A^{\mathcal{O}}_{\mathcal{O}}(y)$:
7 : $\quad\quad$ On $\mathcal{O}$-query x :
8 : $\quad\quad\quad$ **if** $\mathcal{O}(x) = y$:
9 : $\quad\quad\quad\quad$ **abort** emulation with result x
10 : $\quad\quad\quad$ **if** $\mathcal{O}(x) \in I$:
11 : $\quad\quad\quad\quad I \leftarrow I \setminus \{\mathcal{O}(x)\}$
12 : $\quad\quad\quad$ **return** $\mathcal{O}(x)$
13 : $\quad\quad x \leftarrow A^{\mathcal{O}}_{\mathcal{O}}(y) /\!\!/$ or x result of abort
14 : $\quad\quad Z \leftarrow Z \cup \{(x', y) | \mathcal{O}(x') = y, x' \neq x\}$
15 : $\quad$ **endwhile**
16 : $Z \leftarrow Z \cup \{(x', y') |\, y' \notin Y, \mathcal{O}(x') = y'\}$
17 : **return** $(Y, Z, A_{\mathcal{O}})$

$\text{Decoder}(1^\lambda, Y, Z, A_{\mathcal{O}})$

1 : $\mathcal{O} \leftarrow$ Initialize with Z
2 : **while** $Y \neq \emptyset$:
3 : $\quad y \leftarrow \min Y$
4 : $\quad$ Emulate $A^{\mathcal{O}}_{\mathcal{O}}(y)$:
5 : $\quad\quad$ On $\mathcal{O}$-query x:
6 : $\quad\quad\quad$ **if** $\mathcal{O}(x) \neq \perp$:
7 : $\quad\quad\quad\quad$ **return** $\mathcal{O}(x)$
8 : $\quad\quad\quad$ **else**
9 : $\quad\quad\quad\quad$ **abort** emulation with x
10 : $\quad\quad\quad x \leftarrow A^{\mathcal{O}}_{\mathcal{O}}(y) /\!\!/$ or x in abort
11 : $\quad\quad\quad \mathcal{O}(x) \leftarrow y$
12 : $\quad$ **endwhile**
13 : **return** $\mathcal{O}$

Fig. 1. Encoder and Decoder for UROM-oracle $\mathcal{O}$.

query x, the adversary $A_{\mathcal{O}}$ makes to the oracle, we first check if $\mathcal{O}(x) \neq \perp$, i.e., if $\mathcal{O}$ is already defined on that value (line 6). If this is the case, we just return the value saved in $\mathcal{O}$. However, if this is not the case, we know that the call to $\mathcal{O}$ is a hitting query. The reason is that the encoder would have recognized this case and made sure that the value would have been saved in Z (by potentially removing it from I, see line 11) – except for the case where that query is a hitting query. Therefore, in this case, we can already abort the simulation with result x (line 9). If none of the queries is a hitting query and we therefore do not abort, then we eventually obtain x from the adversary (line 10), since the encoder has only put y into Y because the adversary is successful for y. Finally the decoder sets $\mathcal{O}(x)$ to y.

Note that the lexicographic order here is rather arbitrary – the important part is that the encoder always knows exactly which partial information the decoder will have when it will try to decode a specific y, so any fixed order on the images is fine.

The decoder will always return the original oracle $\mathcal{O}$ when given the information the encoder returns. However, we still need to argue that the information returned by encoder is actually smaller than a straightforward description of $\mathcal{O}$.

Lemma 2. *Let $A_{\mathcal{O}}$ be a deterministic adversary against the one-wayness of $\mathcal{O}$ of size $S \leq s(\lambda)$. Further, let $A_{\mathcal{O}}$ be successful on a fraction of $\frac{1}{P}$ of all input challenges $x \in \{0,1\}^\lambda$. With probability $1-2^{-\lambda-1}$ the encoder algorithm describes $\mathcal{O}$ using at most*

$$2\log\binom{2^\lambda}{a} + (2^\lambda - a)\lambda + S$$

bits, where a is defined as $a = \frac{2^\lambda}{n^2 \cdot PS}$.

Proof. First note that with probability $1 - 2^{-\lambda}$, oracle $\mathcal{O}$ will have no y such that y has more than λ^2 preimages. To show this we start with the probability that a specific y has more than λ^2 preimages. For this, we model each of the 2^n inputs x as a random variable X_i such that $X_i = 1$ iff this x maps to y. Then the number of preimages is the sum of all X_i, denoted by X. Now, we can use the Chernoff bound for a binomial distribution $\mathsf{B}(n, p)$ to bound the probability of y having too many preimages:

$$\mathbb{P}\left[X \geq (1+\delta)np\right] \leq \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^{np}.$$

Using $n = 2^\lambda$, $p = 2^{-\lambda}$ and $\delta = \lambda^2$, we get

$$\mathbb{P}_{\mathcal{O}}\left[|\mathcal{O}^{-1}(y)| > \lambda^2\right] \leq \frac{e^{\lambda^2}}{(1+\lambda^2)^{1+\lambda^2}} \leq 2^{-\lambda^2}$$

for $\lambda \geq 3$. Therefore, the probability that each value $y \in \{0,1\}^\lambda$ has at most λ^2 preimages is

$$\mathbb{P}_{\mathcal{O}}\left[\forall y, |\mathcal{O}^{-1}(y)| \leq \lambda^2\right] \geq 1 - 2^\lambda(2^{-\lambda^2}) \geq 1 - 2^{-\lambda-1}.$$

Now that we can assume that the number of preimages of all y is bounded by λ^2, we know that I, the set of all y on which A is successful, has at least size $\frac{2^\lambda}{\lambda^2 \cdot P}$, where $\frac{1}{P}$ is the success probability of $A_{\mathcal{O}}$. Furthermore, $A_{\mathcal{O}}$ makes at most S queries on any input. Hence, for each y the encoder adds to Y, it removes at most S values from I. Therefore, Y has at least size $\frac{2^\lambda}{\lambda^2 \cdot PS}$.

We will now encode Y by giving the positions of the values in Y in $\{0,1\}^\lambda$. For this we need $\log\binom{2^\lambda}{|Y|}$ bits, since we have at most $\binom{2^\lambda}{|Y|}$ such sets of size $|Y|$. Similarly, we can encode the corresponding preimages x in $\{0,1\}^\lambda$ which the encoder found for each $y \in Y$ with the same amount of bits. Denote this set as X. Note that these positions of the x's in X enables a shorter presentation of the set Z of pairs (x', y') with $y' \notin Y$, which the encoder also outputs. Instead of storing the pairs we only need to go through the values $x' \in \{0,1\}^\lambda$ in lexicographic order, skipping over the values in X, and only store the corresponding values y' in this order. This allows us to recover the pairs in Z with the help of the positions in X, but now we only to store $(2^\lambda - |Y|)\lambda$ extra bits to represent Z, plus the $\log\binom{2^\lambda}{|Y|}$ bits to encode X. Finally, we need S bits for the description of $A_{\mathcal{O}}$.

Now, the above size corresponds to the size if the adversary has a success probability of exactly $\frac{1}{P}$ and makes exactly S queries for each input. However, for any sensible parameters, this should yield an upper bound on the size of the

description for any adversary that makes less queries and is successful on a larger fraction of images (if this is not the case, we can of course always adjust our en- and decoder to initialize I with exactly a P-fraction of all ys and always remove exactly S items from I for every y we add to Y). □

Proof (for Theorem 3). To prove Theorem 3, we have to show that for parameters S,P and λ,

$$\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}} \in \mathrm{SIZE}\,(S) : \mathbb{P}_{x\leftarrow\{0,1\}^\lambda}\left[A^{\mathcal{O}}_{\mathcal{O}}(1^\lambda, \mathcal{O}(x)) \in \mathcal{O}^{-1}(\mathcal{O}(x))\right] > \frac{1}{P}\right] < 2^{-\lambda}.$$

Lemma 2 tells us two things: First, that at most a $2^{-\lambda-1}$ fraction of the oracles $\mathcal{O}$ has more than λ^2 preimages for some y. Further, we know that for those oracles with at most λ^2 preimages for each value, if $A_{\mathcal{O}}$ is successful, we can encode the oracle using at most

$$2\log\binom{2^\lambda}{a} + (2^\lambda - a)\lambda + S$$

bits with $a = \frac{2^\lambda}{\lambda^2 \cdot P \cdot S}$. Now, however, this means that the number of oracles that can be encoded in this way is at most

$$\binom{2^\lambda}{a}^2 \cdot 2^{(2^\lambda - a)\lambda} \cdot 2^S.$$

Therefore, using $P, S \le 2^{\lambda/4}$ and $\lambda^2 PS \ge 2$ one can encode only a fraction of

$$\begin{aligned}
\frac{\binom{2^\lambda}{a}^2 2^{(2^\lambda-a)\lambda} 2^S}{2^{\lambda 2^\lambda}} &< \frac{(\frac{e2^\lambda}{a})^{2a}\, 2^S}{2^{a\lambda}} = \frac{e^{2a}2^{\lambda a}2^S}{a^{2a}} = \frac{e^{2a}2^S(\lambda^2 PS)^{2a}}{(2^\lambda)^a} \\
&= 2^S\left(\frac{e^2\lambda^4P^2S^2}{2^\lambda}\right)^{\frac{2^\lambda}{\lambda^2 PS}} < 2^S\left(\frac{e^2\lambda^4P^2S^2}{2^\lambda}\right)^{2^{\frac{\lambda}{2}}} \\
&< 2^{\frac{\lambda}{4}}\left(\frac{e^2\lambda^4 2^{\frac{\lambda}{2}}}{2^\lambda}\right)^{2^{\lambda/2}} < \left(\frac{8\lambda^4 2^{2^{-\lambda/4}} 2^{\frac{\lambda}{2}}}{2^\lambda}\right)^{2^{\frac{\lambda}{2}}} < \left(\frac{8\lambda^4 2^{\frac{\lambda}{2}+1}}{2^\lambda}\right)^{2^{\frac{\lambda}{2}}} \\
&< 2^{-\lambda-1} \text{ for } \lambda \ge 55.
\end{aligned}$$

of all $2^{\lambda 2^\lambda}$ oracles.

In summary, $A_{\mathcal{O}}$ can invert either those oracles $\mathcal{O}$ that have some y with more than λ^2 preimages (which happens with probability at most $2^{-\lambda-1}$), or those that can be encoded as above (which is a fraction of $2^{-\lambda-1}$). Note that both bounds are independent of the choice of S and P. Therefore,the probability that a random oracle $\mathcal{O}$ is invertible with more than probability $\frac{1}{P}$ is bounded by $2^{-\lambda}$:

$$\begin{aligned}
\mathbb{P}_{\mathcal{O}}\left[\exists A_{\mathcal{O}} \in \mathrm{SIZE}\,(s(\lambda)) : \mathbb{P}_{x\leftarrow\{0,1\}^\lambda}\left[A^{\mathcal{O}}_{\mathcal{O}}(1^\lambda, \mathcal{O}(x)) \in \mathcal{O}^{-1}(\mathcal{O}(x))\right] > \frac{1}{P}\right] \\
< 2^{-\lambda-1} + 2^{-\lambda-1} = 2^{-\lambda}.
\end{aligned}$$

This proves the theorem. □

6 Conclusion

In our paper we have presented an alternative approach to define security for idealized hash functions. Whereas the classical random oracle model assumes that the idealized hash function is specific for each adversary, the UROM model allows arbitrary dependencies of the adversary on the random oracle. This appears to be a natural and necessary generalization of the ROM when instantiating the random oracle with known hash functions like SHA-2 or SHA-3. Our UROM has been defined in light of this idea.

Once we had carved out our model, we could evaluate it. We thus related our definition to Unruh's auxiliary-input random oracle model. There, the dependency of the adversary on the random oracle is defined by an unbounded preprocessing stage, giving a polynomial-sized advice to the adversary. We then proved our security notion equivalent to AI-ROM which further solidifies the validity of our UROM definition and, vice versa, also means that the AI-ROM provides strong security guarantees. Remarkably, the security bounds are not tightly related.

One of the differences between the UROM and the AI-ROM, and potentially one of the advantages of the UROM, is that our model allows for more flexibility concerning the sources of insecurities. Specifically, in our model one can separately fine-tune the probabilities for the random oracle and the random choices of the adversary. For instance, one could go so far and simply ask for a non-zero probability for a good random oracle, still stipulating a negligible success probability for the adversary. One could then argue, or hope, that SHA-2 or SHA-3 is indeed one of these good random oracles to provide strong security against all adversaries.

An interesting aspect may be to transfer the UROM or the AI-ROM to the UC setting and the global random oracle model. As mentioned before, the global random oracle model and the idea of having an adversarial dependency on the random oracle (as in UROM and AI-ROM) are incomparable. In principle, however, it should be possible to consider a universal random oracle in the global UC setting as well. Given the subtleties in the simpler game-based setting for defining the UROM, we expect this to be far from trivial, though.

Acknowledgments. We thank the anonymous reviewers for valuable comments. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) − SFB 1119 − 236615297 and by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

A Defining Universal Random Oracles

In this section we present an alternative definition for UROM and argue why it is inappropriate, motivating also our definition of UROM (Definition 3 on page 7).

The Naive Approach. We start with the straightforward adoption of the idea to make the adversary depend on the random oracle by splitting the success probabilities for the experiment Game and the random oracle $\mathcal{O}$, stating that the random oracle should work for all adversaries:

A security game Game is secure in the *naive UROM* if

$$\mathbb{P}_{\mathcal{O}}\left[\begin{array}{c}\forall A_{\mathcal{O}} \in \mathrm{SIZE}\left(\mathrm{poly}(\lambda)\right) \\ \exists \varepsilon_{A,\mathcal{O}} \in \mathrm{negl}\ \forall \lambda\end{array} : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_{A,\mathcal{O}}(\lambda)\right] = 1.$$

We next argue that that there is a game which is trivially insecure when considered in the plain random oracle model, but provably secure according to naive UROM. This is counterintuitive because we expect universal random oracles to provide stronger security guarantees compared to the classical ROM. Let $\mathcal{O}$ be length-preserving and the domain size of the random oracle be $d(\lambda) = \lambda$. The game is defined as:

$$\mathsf{Game}^{A^{\mathcal{O}}_{\mathcal{O}},\mathcal{O}}(=)1 \quad :\Longleftrightarrow \quad \mathcal{O}(0^{\lambda}) \equiv 0 \bmod \lambda^2,$$

where we interpret the λ-bits output of (0^{λ}) as an integer between 0 and $2^{\lambda}-1$. We ignore here for simplicity that this integer reduced modλ^2 is only statistically close to a random number between 0 and λ^2-1, and from now on calculate with a probability of $\frac{1}{\lambda^2}$ that the experiment Game returns 1 and the adversary wins.

First note that this experiment Game is insecure in the standard random oracle mode (Definition 2 on page 7), because the trivial adversary who does nothing wins with non-negligible probability has a success probability of at least $\frac{1}{\lambda^2}$, where the probability is over the choice of $\mathcal{O}$ only. We next show that it is secure in the naive UROM, though. To this end we first negate the security statement of the naive UROM and consider the complementary probability. That is, we have to show:

$$\mathbb{P}_{\mathcal{O}}\left[\begin{array}{c}\exists A_{\mathcal{O}} \in \mathrm{SIZE}\left(\mathrm{poly}(\lambda)\right) \\ \forall \varepsilon_{A,\mathcal{O}} \in \mathrm{negl}\ \exists \lambda\end{array} : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathcal{O}},\mathcal{O}}(\lambda)\right] > \varepsilon_{A,\mathcal{O}}(\lambda)\right] = 0.$$

We first note that the experiment is independent of the adversary, such that we can simplify the statement to:

$$\mathbb{P}_{\mathcal{O}}\left[\forall \varepsilon_{\mathcal{O}} \in \mathrm{negl}\ \exists \lambda : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{\mathcal{O}}(\lambda)\right] > \varepsilon_{\mathcal{O}}(\lambda)\right] = 0.$$

Next observe that the experiment is deterministic, once $\mathcal{O}$ is chosen randomly "on the outside". This means that we can restrict ourselves to negligible functions $\varepsilon_{\mathcal{O}}$ which only take on values 0 and 1, and also drop the probability over Game and instead use the output of the game directly:

$$\mathbb{P}_{\mathcal{O}}\left[\forall \varepsilon_{\mathcal{O}} \in \mathrm{negl}, \varepsilon_{\mathcal{O}} : \mathbb{N} \to \{0,1\}\ \exists \lambda : \ \mathsf{Game}^{\mathcal{O}}(\lambda) > \varepsilon_{\mathcal{O}}(\lambda)\right] = 0.$$

It suffices now to show that, with probability 0 over the choice of $\mathcal{O}$, experiment Game outputs 1 for infinitely many security parameters λ. If the game only

outputs 1 finitely often for a fixed oracle $\mathcal{O}$, say, up to a bound $\Lambda \in \mathbb{N}$, then we can consider the binary-valued negligible function $\varepsilon^{\Lambda}_{\mathcal{O}}(\lambda) = 1$ if $\lambda \leq \Lambda$, and 0 elsewhere. For this function the game's output would not exceed the bound $\varepsilon^{\Lambda}_{\mathcal{O}}(\lambda)$ for any λ. In other words, it suffices to show that the (deterministic) experiment Game outputs 1 for infinitely many security parameters:

$$\mathbb{P}_{\mathcal{O}}\left[\text{for infinitely many } \lambda \in \mathbb{N} : \mathsf{Game}^{\mathcal{O}}(\lambda) = 1\right] = 0.$$

We next apply the Borel-Cantelli lemma to show that this is indeed the case. Let E_λ describe the event that the game is won for security parameter λ. Then $\mathbb{P}_{\mathcal{O}}[E_\lambda] = \frac{1}{\lambda^2}$ over the choice of the random oracle $\mathcal{O}$. Therefore, since the hyperharmonic series converges,

$$\sum_{\lambda=1}^{\infty} \mathbb{P}[E_\lambda] < \infty.$$

The Borel-Cantelli lemma now tells us that the probability that infinitely many E_λ happen is 0. Therefore, the game is indeed secure in the naive UROM.

Towards the Sophisticated UROM. Let us recap what goes wrong with the naive approach above. Borel-Cantelli tells us that for a random oracle $\mathcal{O}$ the probabilities of Game outputting 1 become small such that the adversary will only be successful on finitely many security parameters (with probability 1). This yields a fundamental, yet from a cryptographic perspective somewhat counterintuitive property of adversaries: *An adversary might be only successful on finitely many security parameters (except with probability* 0*), even though the adversary has a polynomial success probability for each individual security parameter!*

The difference to the ordinary random oracle model is that, there, we rather state security in reverse order, i.e., for a given security parameter λ the probability of an adversary breaking the game for random oracle $\mathcal{O}$ is negligible. We would like to resurrect this behavior while preserving the idea of having a universal random oracle. The approach is basically to move out the quantification over all security parameters ($\forall\lambda$) out of the probability for oracle $\mathcal{O}$. This, however, means that the preceding quantification over the adversary and the negligible function ($\forall A \exists \varepsilon \forall \lambda$) needs to be moved outside of $\mathbb{P}_{\mathcal{O}}[\cdot]$ as well. But this infringes with our idea of the universal random oracle model where the adversary may depend on $\mathcal{O}$. To re-install this property we only move out a bound $s(\lambda)$ on the adversary's size, and still quantify over all adversaries of this maximal size $s(\lambda)$. This yields our definition of the universal random oracle model (Definition 3):

$$\forall s \in \text{poly}\ \exists \varepsilon_s \in \text{negl}\ \forall \lambda \in \mathbb{N} :$$
$$\mathbb{P}_{\mathcal{O}}\left[\forall A_{\mathcal{O}} \in \text{SIZE}(s(\lambda)) : \mathbb{P}_{\mathsf{Game}}\left[\mathsf{Game}^{A^{\mathcal{O}}_{\mathcal{O}},\mathcal{O}}(\lambda)\right] \leq \varepsilon_s(\lambda)\right] \geq 1 - \varepsilon_s(\lambda). \quad (4)$$

The outer negligible function $\varepsilon_s(\lambda)$ now becomes necessary since for fixed λ we only consider oracle $\mathcal{O}$ of restricted input and output size, determined by the size bound of the adversary and the fixed game.

Besides the equivalence to the auxiliary-input random oracle model and the immediate implication that security in this version of the UROM implies security for ordinary random oracles, we can also discuss directly why our counter example for the naive approach is also labeled as insecure. Recall that $\mathsf{Game}^{\mathcal{O}}(\lambda)$ outputs 1 if $\mathcal{O}(0^\lambda) \equiv 0 \bmod \lambda^2$. Then for any given parameter λ we have $\mathbb{P}_{\mathcal{O}}\left[\mathsf{Game}^{\mathcal{O}}(\lambda) = 0\right] \leq 1 - \frac{1}{\lambda^2}$. It follows that there is no negligible bound $\varepsilon_s(\lambda)$ such that this probability at least $1 - \varepsilon_s(\lambda)$.

References

1. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_11
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 1993, pp. 62–73. ACM Press (1993). https://doi.org/10.1145/168588.168596
3. Buldas, A., Laur, S., Niitsoo, M.: Oracle separation in the non-uniform model. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 230–244. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04642-1_19
4. Buldas, A., Niitsoo, M.: Black-box separations and their adaptability to the non-uniform model. In: Boyd, C., Simpson, L. (eds.) ACISP 2013. LNCS, vol. 7959, pp. 152–167. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39059-3_11
5. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 280–312. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_11
6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press (2001). https://doi.org/10.1109/SFCS.2001.959888
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004). https://doi.org/10.1145/1008731.1008734
8. Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014, pp. 597–608. ACM Press (2014). https://doi.org/10.1145/2660267.2660374
9. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_9
10. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_16
11. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12

12. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: 41st FOCS, pp. 305–313. IEEE Computer Society Press (2000). https://doi.org/10.1109/SFCS.2000.892119
13. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge (2001)
14. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS, pp. 102–115. IEEE Computer Society Press (2003). https://doi.org/10.1109/SFCS.2003.1238185
15. Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. SIAM J. Comput. **44**(1), 193–242 (2015). https://doi.org/10.1137/130938438
16. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61. ACM Press (1989). https://doi.org/10.1145/73007.73012
17. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_8
18. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000). https://doi.org/10.1007/s001450010003
19. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_12