Augmented Random Oracles

Mark Zhandry
NTT Research & Princeton University
mzhandry@gmail.com

Abstract

We propose a new paradigm for justifying the security of random oracle-based protocols, which we call the Augmented Random Oracle Model (AROM). We show that the AROM captures a wide range of important random oracle impossibility results. Thus a proof in the AROM implies some resiliency to such impossibilities. We then consider three ROM transforms which are subject to impossibilities: Fiat-Shamir (FS), Fujisaki-Okamoto (FO), and Encrypt-with-Hash (EwH). We show in each case how to obtain security in the AROM by strengthening the building blocks or modifying the transform.

Along the way, we give a couple other results. We improve the assumptions needed for the FO and EwH impossibilities from indistinguishability obfuscation to circularly secure LWE; we argue that our AROM still captures this improved impossibility. We also demonstrate that there is no "best possible" hash function, by giving a pair of security properties, both of which can be instantiated in the standard model separately, which cannot be simultaneously satisfied by a single hash function.

1 Introduction

The random oracle model (ROM) [BR93] treats a cryptographic hash function as a random function, and is a crucial tool for analyzing the security of cryptosystems that otherwise lack a "standard model" security proof. This model captures most practical cryptographic techniques and attacks involving hash functions. Constructions with ROM proofs are often far more efficient than their standard-model counterparts, and numerous applied cryptosystems utilize this model.

Unfortunately, there are numerous examples of ROM failures, schemes that have been proven secure in the ROM but are insecure when the hash function is instantiated. Starting with [CGH98], the most problematic such failures are *uninstantiability* results, where the protocol is insecure under *any* instantiation of the hash function. This makes it challenging to understand the meaning of a ROM proof, and has lead to significant debate (see e.g. [Gol06, KM15]). Nevertheless, due to their efficiency, schemes with only ROM proofs remain widely deployed.

This practice is often justified by observing that ROM uninstantiabilities are typically contrived, deviating from standard cryptographic design. However, there are also examples of *natural* uninstantiabilities, even those for design structures widely used in practice, though this has never lead to actual real-world attacks. We will discuss several examples later in this work. In light of this state-of-affairs, it is important to further understand the security of ROM protocols.

Techniques for uninstantiability results. Digging deeper, all known ROM uninstantiability results make essential use of *non-black-box* techniques. They use that real hash functions have

code which can be plugged into tools like proof systems, fully homomorphic encryption, program obfuscation, etc. Random oracles, by contrast, cannot be plugged into such tools as they have no code. The ROM uninstantiabilities therefore embed a trigger that can only be accessed by feeding the hash function code into such a tool; this trigger completely breaks security.

More generally, even when considering non-black box tools, essentially all cryptographic techniques use the component systems as black boxes. Even though non-black box tools take programs as input, the programs themselves only treat the component as a black box. The application of these tools does not care about the actual code of components, other than the fact that it has code in the first place. Of course, the implementation of the non-black-box tool will operate on the actual code at the gate or instruction level, but the tool abstracts all this away. The application of the tool only cares that the code exists.

1.1 Augmented Random Oracles

In this work, with the goal of eliminating uninstantiability results, we propose a new paradigm for studying ROM constructions that we call the *Augmented* Random Oracle Model (AROM). In addition to a random oracle O, we add a second oracle M, which will model the various non-black-box tools that ROM impossibilities may try to employ. Like O, M will be a function sampled from a distribution[1]. However, to model tools that can be applied to the code of concrete hash function (which is now an oracle), we will have M be *oracle aided*, meaning it can make queries to O. Making queries is the only way M can learn information from O. Looking ahead, we will often have M take as input programs that themselves query O; M can then evaluate such programs by making queries to O. In this way, we can treat O as having code—namely the instruction to make a query—while still representing O as an oracle, thus capturing the aforementioned non-black-box techniques within our idealized model.

Asharov and Segev [AS15] consider a similar model, but for an entirely different purpose. They propose a model for indistinguishability obfuscation (iO), where M obfuscates programs that can make queries to O. Such M accepts `obfuscate` queries, which take as input the description of an (oracle-aided) program P^O, and outputs a string $\tilde{P}$, derived via a private random permutation. M also accepts `evaluate` queries, which take $\tilde{P}$ and an input x, and compute $P^O(x)$. Note that M must make queries to O in order to implement `evaluate` queries. The authors argue that this model captures many of the techniques for using iO. [AS15] use this model to reason about the limits of iO.

As this oracle captures many techniques based on iO, setting M in this way would capture many uninstantiability results based on iO, such as [BFM15] as discussed below. However, we do not want to commit to a single tool. This is for several reasons. The Asharov-Segev model, for example, makes specific choices, such as the fact that M does not apply to programs that themselves can make M queries, or that M operates on oracle-aided *circuits* as opposed to Turing machines. There are also many other non-black-box tools such as proof systems, garbled circuits, fully homomorphic encryption, etc. Asharov and Segev specifically mention the case of NIZKs, as many iO applications involve NIZKs but are not captured in their model. Worse, new non-black-box tools may arise, necessitating new models. We therefore allow M to be *any* oracle, which automatically captures any tool of this nature and any modeling that may arise. In this sense, we can think of M *adversarially*.

[1]Once M is sampled, it is fixed an immutable, keeping no state. Though M is stateless, it can still implement potentially stateful cryptographic objects, by having any state be an explicit input and output of M. Modeling M as stateless reflects the real world, where the specification of a cryptographic primitive does not change over time.

We will make one important restriction, however: M can only make a polynomial number of queries to O, corresponding to the tool being efficient.

On the other hand, we do not want to rely these black-box tools when designing cryptosystems. First, they are computationally expensive. Moreover, since M is essentially adversarial, we do not want to have to assume any particular structure of M; M could always output 0. We will therefore insist that the system we design, and hence also the security game, only makes queries to O but not M. Thus we only consider constructions that make sense in the plain random oracle model, but we hope that the new model will better capture their security. A visualization of the plain and augmented models are given in Figure 1.

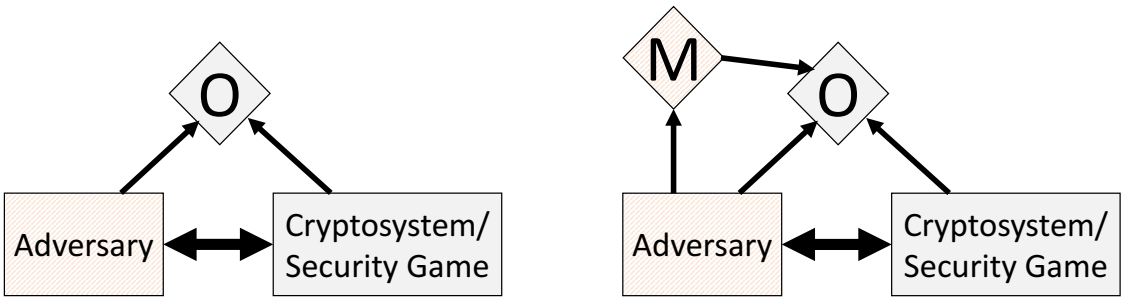

Figure 1: The plain ROM (L) vs the Augmented ROM (R).

So far, the AROM appears rather useless: by having the adversary simulate M for itself, the AROM collapses to the standard ROM. We will now see an important setting where the AROM is meaningful.

AROM for Transforms. In a ROM *transform*, a building block Π (or potentially multiple building blocks) is transformed into a different cryptosystem Γ using a random oracle. We note ROM transforms are widespread, and even schemes that appear to be direct constructions can often be phrased as transforms from appropriate abstractions. Examples include RSA (trapdoor permutations) or Diffie-Hellman (cryptographic groups). Moreover, uninstantiability results such as [CGH98] are most often phrased as transforms from the appropriate building blocks (e.g. CS proofs [Mic94] and signatures in the case of [CGH98]). In fact, phrasing a construction as a transform is generally the *preferred* way to model constructions, as it allows for utilizing abstractions, resulting in a better conceptual understanding of the results and more general security proofs.

Well-known uninstantiabilities for ROM transforms include Fiat-Shamir (FS) [FS87] as proved by [GK03], and Encrypt-with-Hash (EwH) [BBO07] and Fujisaki-Okamoto (FO) [FO99], as shown by [BFM15]. These ROM failures for transforms are notable for being for *natural* and even widely deployed.

For transforms, the picture in Figure 1 changes. Recall that a transform must result in a secure Γ, regardless of the instantiation of the building block Π, as long as Π satisfies the prescribed security property. Since the security of the transform quantifies over all Π, we can think of Π itself as adversarial. ROM transform uninstantiabilities work exactly by designing a contrived Π that makes the transform fail. In the plain ROM, this gives Figure 2 (L). In the AROM, the transform still only queries O, but now Π may use M to employ non-black box techniques. Therefore, Π makes queries to M, as in Figure 2 (R).

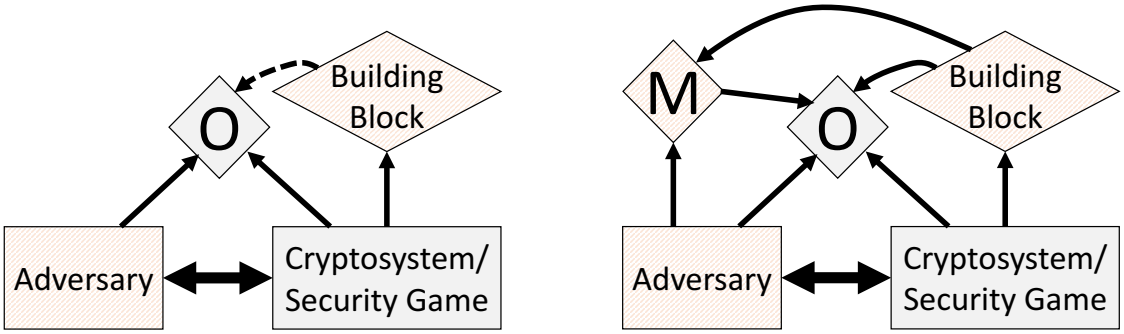

Figure 2: Plain ROM transforms (L) vs Augmented ROM transforms (R). Some authors model the building block as having access to O while others do not; in Section 1.4 we argue that the best modeling would give access to O, and so we adopt this convention in the AROM.

Now we see that the AROM is not trivially equivalent to the ROM. Concretely, the security of the building block may rely on the fact that M is sampled from a distribution. For example, the M above for implementing iO is only secure because the obfuscator utilizes a private random permutation. In order to maintain security, this permutation must be hidden from the adversary. Therefore, there is no way for the adversary to simulate the M on its own. Indeed, we will argue that the AROM captures all existing uninstantiability results for ROM transforms, by having M model the appropriate non-black box techniques. This does not mean that the AROM is not subject to uninstantiability results, since it is equivalent to the ROM for direct constructions. More generally, one can take any uninstantiability result, even for transforms, and instantiate the building blocks with particular constructions from the literature, arriving at a direct ROM uninstantiability result, which would then also be an AROM uninstantiability.

However, suppose we have a transform that is *fully abstracted*, in the sense that any cryptography being performed is abstracted underneath an appropriate building block that is input to the transform[2]. Then we argue that all known uninstantiability results for random oracles are captured by the AROM, in the sense that, if fully abstracted, the transform would be correctly labeled as insecure in the AROM. This is because, for any such result, there will be an M which can securely provide all the necessary building blocks, but also the non-black box techniques used, where we replace any time the code of the hash function is used with the instruction to query O. This includes [CGH98] and also the uninstantiability for FS [GK03], where M implements CS proofs on programs that can query O. It also includes the uninstantiability of EwH and FO [BFM15], where M implements an indistinguishability obfuscator. In fact, for any known uninstantiability of the random oracle, when fully abstracted, there is an appropriate M that models the building blocks, resulting in an insecure protocol in the AROM. See Section 4.3, where we work through the case of EwH.

Thus, for any fully abstracted protocol, security in the AROM demonstrates immunity to known

[2]We do not attempt formalize *full abstracted* here, as it appears challenging. Do information-theoretic objects, such as even a simple XOR, count? We instead leave the notion as a general intuitive property, and we expect that whether or not a given protocol is fully abstracted will usually be clear. All the transforms we consider in this work are certainly fully abstracted.

uninstantiability techniques, and offers the most compelling evidence known for real-world security. Of course, this does not actually prove security in the standard model or completely rule out uninstantiability results, but it implies that brand new techniques would be needed to invalidate security.

1.2 Best Possible Hash Functions?

A quick detour before getting to our results. There have been numerous works on circumventing ROM impossibilities, or at least making ROM proofs more believable. Here, we discuss one, initiated by Canetti [Can97], which seeks to identify and instantiate random oracle security properties using concrete, usually algebraic, hash functions. Examples include oracle hashing [Can97], non-malleable point obfuscation [KY18, BMZ19], various forms of correlation resistance [CCR16, GOR11], and Full Domain Hash [HSW14], to name a few.

A major downside of these results is efficiency. In essentially all cases, the construction is far less efficient than standard hash functions such as SHA2, sometimes being entirely impractical. In addition, the computational assumptions underlying these ROM-free constructions can be quite strong, and it is not clear if the standard model result is actually "more secure."

In light of these downsides, a standard-model instantiation of a ROM protocol may be considered a *proof of concept*, showing that such an application is likely to exist. This could be seen as additional justification for the security (or at least, lack of impossibility) for the more efficient ROM protocol. Implicit in this interpretation is the following assumption: if a security property holds for *some* hash function, then it also holds for a sufficiently well-designed hash function, perhaps SHA2. That is, SHA2 is a "best possible" hash function, in that any security property which holds for *some* hash function will hold for SHA2 [3]. This sounds plausible, even in light of the various ROM impossibility results, as no poly-time attacks have been found on SHA2 that does not also apply to all hash functions. We ask, *is such an interpretation reasonable?*

1.3 Our Results

- In Section 3 we formally define the AROM.
- We then use the EwH transform as a case study to demonstrate the power of the AROM. In Section 4.3, we explain how the AROM captures the uninstantiability of EwH, in the sense that the transform is *insecure* in the AROM, like in the real world.
- In Section 4.4, we show the EwH uninstantiability result can be generalized to work under a circular security assumption on LWE, as opposed to needing the full power of indistinguishability obfuscation. Concretely, our impossibility uses fully homomorphic encryption and obfuscation for compute-and-compare programs [GKW17, WZ17]. The improvement also readily adapts to the FO transform. This further demonstrates the need for a model which captures a variety of non-black-box tools.
- In Section 4.6, we show that EwH *is* secure in the AROM, if the underlying encryption scheme is strengthened to be *lossy* [BHY09]. Lossy encryption can still be constructed efficiently from most standard tools. We note that the security we prove likely cannot be proven secure in the

[3] There will always be *functionalities* that SHA2 or other hash functions cannot achieve. This assumption is only about *security properties* that apply to any hash function.

standard model [Wic13], so some form of idealized model is inherent. Our proof offers the strongest justification yet for security.

- We next study the FO and FS transformations, demonstrating that both are insecure in the AROM, again capturing the known uninstantiabilities. For FS, we show that it is sound in the AROM if the underlying proof has statistical soundness. Like EwH, FS even for such proofs likely cannot be proven secure in the standard model [BDG+13], necessitating some idealized model. Our proof offers the strongest justification yet for security in this case. We note that zero knowledge of plain Fiat-Shamir cannot be proved, since this would give NIZKs without a CRS. We explore several ways of obtaining zero knowledge by introducing a CRS.

  For FO, we observe that it is *not* secure in the AROM, even if the underlying encryption scheme is lossy. We therefore propose (Section 5.1) a new encryption scheme, which can be seen as a variant of the CCA-secure scheme of Dolev, Dwork, and Naor [DDN91], but with the zero knowledge proof replaced by an EwH-style structure. We prove CCA security of our scheme under the assumed lossiness of the underlying encryption scheme; CCA security is not known to follow from lossy encryption in the standard model.

  Our results for FO and FS are given in Sections 5 and 6.

- In Section 7, we provide a pair of natural security properties for hash functions, namely auxiliary input one-wayness and something we call *anti-lossiness*. These properties can be satisfied by standard-model constructions, and are both trivially satisfied by random oracles. However, we show that these properties *cannot* both be satisfied simultaneously by any real hash function, assuming virtual grey box (VGB) obfuscation [BCKP14]. This implies that SHA2 (or any hash for that matter) cannot be a "best possible" hash.

  In the AROM, only one of the two properties—namely anti-lossiness—hold for O, consistent with the standard model. This gives further support to the utility of our model, and also indicates that SHA2 (or any hash function plausibly modeled as a random oracle) is likely not auxiliary input one-way.

1.4 A Classification of ROM Failures

Besides uninstantiability results, there are a number of other known ROM failures. Here, we broadly organize known ROM failures into five types, and discuss what they mean and their relevance to the AROM.

Type 1 ($\exists\exists$)**.** Here, there exists a specific protocol with a ROM proof and also a specific hash function H, such that setting $O = H$ makes the protocol insecure.

A well-known example is the length-extension attack when using Merkle-Damgård as MACs without appropriate padding. Another example is the circularly secure encryption scheme $\mathsf{Enc}(k, m) = (r, O(k, r) \oplus m)$, which was proven in the ROM [BRS03], but is insecure when O is replaced with Davies-Meyer [HK07].

For Type 1 failures, the insecurity may point to an issue with the protocol, the hash, or both. However, we observe that in most cases, the particular hash function is not *indifferentiable* [MRH04] from a random oracle (see [CDMP05] for Merkle-Damgård, [KM07] for Davies-Meyer). Indifferentiability has become an important consideration for hash functions, and so an indifferentiability failure

should be interpreted as a weakness of the hash function. In particular, using an indifferentiable hash function seems to solve the problem.

More generally, any Type 1 failure will point to a hash function design structure that, if avoided, would block the attack. Such a design structure may then be considered sub-optimal from a security standpoint.

Type 2 ($\forall\exists$)**.** Here, for *any* possible hash function H, there exists a protocol with a ROM proof such that setting $O = H$ makes the protocol insecure.

Type 2 failures were already pointed out by [BR93]. For a typical example, consider the Encrypt-with-Hash (EwH) transform $\mathsf{Enc}'(\mathsf{pk}, m) = \mathsf{Enc}(\mathsf{pk}, m;\ O(\mathsf{pk}, m)\ )$ which converts a randomized public key encryption scheme into a deterministic one by setting the random coins to $O(\mathsf{pk}, m)$ [BBO07]. For any concrete hash function H, there is an Enc that renders the transform trivially insecure when $O = H$: $\mathsf{Enc}(\mathsf{pk}, m; r)$ checks if $r = O(\mathsf{pk}, m)$ and if so outputs m in the clear.

For Type 2 failures, we observe that the ROM security is an artifact of the ROM modeling: concretely, when [BBO07] prove ROM security, they assume that Enc cannot make queries to O. But certainly a real-world encryption scheme may evaluate a given hash function. In fact, since there are a limited number of standardized hash functions, it is even expected that different components of a cryptosystem may use the same hash. So a better modeling would allow Enc to query O, in which case EwH is trivially insecure in the ROM for the same reasons as in the standard model. Therefore, Type 2 failures can be seen as demonstrating an issue with the particular protocol design, but not the random oracle itself if properly modeled. Instead, it shows that the scheme should never have been considered to have a ROM proof in the first place.

We observe that our AROM *always* allows the building block to query O (since M may implement a query-forwarding functionality), so failures of this sort are captured by the AROM, in the sense that such protocols will not have AROM proofs. We note that a tweaked EwH, namely $\mathsf{Enc}'(\mathsf{pk}', m) = \mathsf{Enc}(\mathsf{pk}, m;\ O(s, m)\ )$ for $\mathsf{pk}' = (\mathsf{pk}, s)$ and a uniformly random s *would* be secure in the ROM, even if Enc can make random oracle queries. The reason, essentially, is that the random s enforces domain separation, since Enc would almost certainly never evaluate O on inputs of the form (s, m). Nevertheless, the impossibility of [BFM15] still applies to the tweaked EwH.

Type 3 ($\exists\forall$)**.** Here, there exists a protocol with a ROM proof that is insecure under *any* possible instantiation of the hash function.

These are the uninstantiability results motivating our AROM. As observed above, for fully abstracted transforms, no known Type 3 failures apply to the AROM.

Type 4 (Simulation-based)**.** Here, security is defined via a simulator, and in the ROM the simulator is allowed to program the random oracle.

Examples include non-interactive zero knowledge without a CRS [Pas03] and non-interactive non-committing encryption [Nie02], both of which exist in the ROM under this modeling of simulators, but not in the real world. The intuition for these failures is that, in the standard model, the simulator is usually required to have extra power relative to the adversary — such as being able to program a CRS or generate transcript messages out of order — in order to not be trivially impossible. Since the adversary cannot program the random oracle, allowing the simulator such programming ability is another form of extra power, allowing it to circumvent standard-model

impossibilities without having to resort to CRS's or out-of-order transcript generation. This allows for attainable simulation-based definitions that are impossible in the standard model.

One problem with Type 4 failures is that the random oracle is baked into the security definition since the definition must model the simulator's ability to program the random oracle. This makes the ROM definition actually distinct from the standard model definition. Failures of this type are typically easily avoided by better modeling of the ROM: allow the simulator to make random oracle queries, and even see the adversary's queries, but *do not* allow the simulator to actually program the random oracle. The resulting definition then closely mirrors the standard model, and the only options available to give the simulator the needed extra power are generally the same strategies as in the standard model. For these reasons, we advocate similar modeling of simulators in the AROM.

Type 5 (Proof impossibilities)**.** Here, it is proved that, for some protocol with a ROM proof, there cannot be any standard-model proof relative to any hash function, at least with respect to certain classes of constructions, proof strategies, and/or underlying computational assumptions.

A well-known example is Full-Domain Hash (FDH), where [DOP05] show that there is no proof of security in the standard model that makes fully black box use of the trapdoor permutation. A wide class of examples of this type are impossibilities of security proofs relative to "falsifiable" assumptions. Examples include Fiat-Shamir *even when restricted to statistically sound proofs*[4] [BDG+13], succinct non-interactive arguments (SNARGs) [GW11], and correlated input security [Wic13]. We note that correlated input security is in particular implied by the notion of security we prove in the AROM for EwH.

With Type 5 examples, no actual insecurity is shown, just a barrier to proving security. It could therefore be that the examples are in fact secure, but just cannot be demonstrated secure by standard model arguments. An optimistic interpretation is that such examples are actually demonstrating limits of the usual paradigm for provable security, with the ROM offering a way to plausibly justify the security of such protocols. However, in light of Type 3 failures, a pessimistic interpretation could simply be that Type 5 examples are simply insecure. The right answer probably lies somewhere between.

Nevertheless, protocol designs subject to Type 5 failures have been confidently used in practice, such as Fiat-Shamir (not to mention FDH and SNARGs). It is therefore important to try to justify their security despite such Type 5 failures. We can therefore view the AROM as offering additional support for the security of such schemes. This is particularly relevant for our AROM proofs of EwH and Fiat-Shamir for statistically sound proofs, as a standard-model security justification is infeasible.

1.5 Discussion: Do we really need another ROM variant?

There have been many attempts to rectify the issues with the ROM, each with their own advantages as disadvantages. Numerous works remove the random oracle entirely from a cryptosystem, such as Boneh and Boyen [BB04] for IBE. But such results typically lose significant efficiency, and sometimes require stronger assumptions as well. The aforementioned program initiated by Canetti [Can97] shows how to instantiate certain ROM properties, but likewise results in inefficient hash functions and often requires strong assumptions. One might be tempted to use such results as proofs of

[4]The Type 3 counterexample of [GK03] uses computationally sound protocols.

concept and then just use SHA2 to instead, but our results on incompatible security properties show that this is unsound in general.

Both programs also suffer from the fact that they cannot bypass certain limitations of the standard-model, such as the Type 5 ROM failures discussed above. In order to justify the security of these examples, *something* is needed beyond the standard model.

Another approach is to identify a broad class of standard-model security notions, and posit that a hash function simultaneously satisfies the entire class. One example are universal computational extractors (UCEs) [BHK13]. However, it appears challenging to define a natural broad class of security notions that are exempted from ROM failures. In particular, the UCE assumption of [BHK13] is subject to the Type 3 failure from EwH.

This leaves other refinements to the ROM. Nielsen [Nie02] defines the *non-programmable* ROM (npROM) where the reduction is prevented from programming O in any way, but can still see the queries the adversary makes. The hope is that this more closely captures standard-model hash functions "behaving like" random functions, since standard-model functions cannot be programmed. A complementary model due to Ananth and Bhaskar [AB12] is the *non-observable* ROM (noROM), where the adversary can adaptively program but cannot observe. They also consider the intersection of the two models, the nonpROM.

These refinements are intuitively appealing. But there is little theoretical justification for preferring them over the plain ROM. Type 3 failures also still apply: the CPA security of EwH [BBO07] can be proven in the nonpROM, and yet we know the transform is insecure in general [5].

Our model, by contrast, is specifically shown to circumvent all known Type 2, 3, and 4 failures for fully abstracted transforms, and the other failures can be handled by using a sufficiently well-designed hash function and making optimistic-yet-plausible assumptions. We thus obtain some of the most compelling justifications known for several cryptosystems.

2 Preliminaries

2.1 Cryptosystems and games

A cryptosystem is a tuple of stateless deterministic algorithms Π. A *specification* for a cryptosystem is a collection $\mathcal{G}$ of game/probability pairs (G, p), where G take a security parameter 1^λ as input and outputs a bit b, and p takes a security parameter 1^λ as input and outputs a real number in $[0, 1]$. Each G interacts with a cryptosystem Π and adversary $\mathcal{A}$. We also assume G indicates whether adversaries are computationally bounded or unbounded. We will write $b \leftarrow (\mathcal{A} \leftrightarrow G^\Pi)(1^\lambda)$ to denote the interaction. The *advantage* of $\mathcal{A}$ when interacting with G^Π is a function of λ defined as $\mathsf{Adv}_{\mathcal{A},G^\Pi}(\lambda) := \Pr[1 \leftarrow (\mathcal{A} \leftrightarrow G^\Pi)(1^\lambda)] - p(\lambda)$. Games model both security properties and correctness properties.

Many cryptosystems will use random coins, which we model as an explicit input. Games will be responsible for choosing the random coins. We will often distinguish random coins from other inputs by separating them with a semicolon, e.g. $\Pi(x; r)$. We will write $\Pi(x)$ to be the distribution $\Pi(x; r)$ for uniform r. A function is *negligible* if it is asymptotically smaller than any inverse polynomial.

Definition 2.1. A cryptosystem Π *securely implements* a specification $\mathcal{G}$ if, for all $(G, p) \in \mathcal{G}$ and for all adversaries $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}_{\mathcal{A},G^\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

[5] Bellare et al. [BBO07] do not claim either noROM or npROM. Yet the proof in the CPA case can be verified to work in both models by inspection.

Transforms. A transform is a method T of compiling a cryptosystem Π securely implementing a specification $\mathcal{G}$ into another cryptosystem Γ securely implementing a specification $\mathcal{H}$. We write $\Gamma = T^{\Pi}$.

Definition 2.2. A transform T from $\mathcal{G}$ to $\mathcal{H}$ is secure if, for all Π which securely implement $\mathcal{G}$, T^{Π} securely implements $\mathcal{H}$.

Single-stage games. Usually, $\mathcal{A}$ is a single adversary that can keep arbitrary state throughout its interaction with $\mathcal{G}$. We will call these single-stage games. Some games place restrictions on the state $\mathcal{A}$ can keep. We call such games *multi-stage*.

2.2 Cryptographic Definitions

An $\ell = \ell(\lambda)$-source is a distribution is a family of efficiently sampleable distributions $D(1^\lambda)$ over tuples $(x_1, \ldots, x_\ell, \mathsf{aux})$.

Definition 2.3 (Unpredictability)**.** A 1-source $(x, \mathsf{aux}) \leftarrow D(1^\lambda)$ is *computationally (resp. statistically) unpredictable* if, for all polynomial time (resp. unbounded) $\mathcal{A}$, $\Pr[\mathcal{A}(\mathsf{aux}) = x : (x, \mathsf{aux}) \leftarrow D(1^\lambda)]$ is negligible.

An ℓ-source ($\ell > 1$) is *computationally (resp. statistically) unpredictable* (1) if each marginal distribution (x_i, aux) for $i \in [\ell]$ is computationally unpredictable, and (2) except with negligible probability the x_i are all distinct.

Definition 2.4 (Anti-lossiness)**.** A keyed function $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ is *anti-lossy* if, for all sequences $(k_\lambda)_\lambda$ for $k_\lambda \in \{0,1\}^\lambda$, the 1-source $(H(k_\lambda, x), \mathsf{aux} = \{\})$ where $x \leftarrow \{0,1\}^{m(\lambda)}$ is statistically unpredictable. In other words, there are no keys which make H lose too much information.

Definition 2.5 (One-wayness with correlated inputs)**.** A keyed function $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ is *one-way against correlated inputs* if, for all computationally unpredictable ℓ-sources D and all polynomial-time $\mathcal{A}$,

$$\Pr\left[\exists i, H(k, x') = y_i : \begin{array}{c} k \leftarrow \{0,1\}^\lambda \\ (x_1, \ldots, x_\ell, \mathsf{aux}) \leftarrow D \\ x' \leftarrow \mathcal{A}(k, y_1 = H(k, x_1), \ldots, y_\ell = H(k, x_\ell), \mathsf{aux}) \end{array}\right] < \mathsf{negl}(\lambda) \ .$$

That is, given aux and all the $y_i = H(k, x_i)$, it is intractable to invert any of the y_i. H is *one-way against auxiliary input* if the above holds only for 1-sources.

Definition 2.6 (Pseudorandomness with correlated inputs)**.** A keyed function $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ is *pseudorandom against correlated inputs* if, for all computationally unpredictable ℓ-sources and all polynomial-time $\mathcal{A}$,

$$\left|\Pr\left[b' = b : \begin{array}{c} b \leftarrow \{0,1\}, k \leftarrow \{0,1\}^\lambda \\ (x_1, \ldots, x_\ell, \mathsf{aux}) \leftarrow D \\ y_{i,0} \leftarrow H(k, x_i), y_{i,1} \leftarrow \{0,1\}^{n(\lambda)} \forall i \\ b' \leftarrow \mathcal{A}(k, y_{1,b}, \ldots, y_{\ell,b}, \mathsf{aux}) \end{array}\right] - 1/2\right| < \mathsf{negl}(\lambda) \ .$$

In other words, the vector of $y_i = H(k, x_i)$ is pseudorandom, even though the x_i are correlated and aux is given. H is *pseudorandom against auxiliary input* if the above holds only for 1-sources.

Public key encryption (PKE). A PKE scheme is a triple $\Pi = (\mathsf{Gen}, \mathsf{Dec}, \mathsf{Enc})$ such that $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) = \mathsf{Gen}(1^\lambda; r)$, $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m) = \mathsf{Enc}(\mathsf{pk}, m; r)$ and $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. We require *correctness*, which insists that for every message m, $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)] \geq 1 - \mathsf{negl}(\lambda)$.

Definition 2.7 (CPA and CCA security)**.** A PKE scheme Π is *CCA secure* if all polynomial time $\mathcal{A}$ have negligible advantage in the following game:

- On input 1^λ, the game samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
- $\mathcal{A}$ makes CCA queries on ciphertexts c, and receives $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$.
- At some point, $\mathcal{A}$ produces two messages $m_0^*, m_1^* \in \{0,1\}^*$ of equal length.
- The game samples a random bit b and replies with $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*)$.
- $\mathcal{A}$ can continue making CCA queries, as long as $c \neq c^*$.
- $\mathcal{A}$ finally sends a guess b' for b. The advantage of $\mathcal{A}$ is $|\Pr[b' = b] - 1/2|$.

Π is CPA secure if the above only holds against $\mathcal{A}$ that cannot make CCA queries.

Definition 2.8 (Lossy Encryption [BHY09])**.** A PKE scheme Π is *lossy* if there is an additional algorithm $\mathsf{pk} \leftarrow \mathsf{GenLossy}(1^\lambda)$ such that:

- $\mathsf{pk} \leftarrow \mathsf{GenLossy}(1^\lambda)$ is comp. indist. from pk where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- Let D_m be the distribution $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m))$ where $\mathsf{pk} \leftarrow \mathsf{GenLossy}(1^\lambda)$. Then for any messages m, m' of the same length, $D_m, D_{m'}$ are statistically close.

Definition 2.9 (Fully Homomorphic Encryption)**.** A PKE scheme Π is fully homomorphic if there is an additional algorithm $\mathsf{Eval}(\mathsf{pk}, c, f)$ that outputs ciphertexts, such that for all m and all functions f represented as circuits, the following hold:

$$\mathsf{length}(\ \mathsf{Eval}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m), f)\ ) = \mathsf{length}(\ \mathsf{Enc}(\mathsf{pk}, f(m))\ ), \text{ and}$$

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c') = f(m) : \begin{smallmatrix}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ c' \leftarrow \mathsf{Eval}(\mathsf{pk}, c, f)\end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\lambda)\ .$$

Deterministic Encryption. A deterministic PKE scheme is plain PKE, except that Enc is deterministic. Deterministic PKE can only be secure for unpredictable messages, formalized by PRIV security [BBO07]:

Definition 2.10 (PRIV-CPA and PRIV-CCA)**.** A det. PKE scheme Π is strongly (resp. weakly [6]) *PRIV CCA secure* if for all computationally (resp. statistically) unpred. ℓ-sources D, all polynomial time $\mathcal{A}$ have negligible advantage in the following game:

- On input 1^λ, the game samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
- It samples $(m_{1,0}^*, \ldots, m_{\ell,0}^*) \leftarrow D$ and random *distinct* $m_{1,1}^*, \ldots, m_{\ell,1}^*$.

[6] The original PRIV notion corresponds to the weak version.

- It samples a random bit b, and sends $c_1^*, \ldots, c_\ell^*$ where $c_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{i,b}^*)$.
- $\mathcal{A}$ makes CCA queries on $c \notin \{c_1^*, \ldots, c_\ell^*\}$; it receives $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$.
- $\mathcal{A}$ finally sends guess b' for b. The advantage of $\mathcal{A}$ is $|\Pr[b' = b] - 1/2|$.

Π is strongly/weakly PRIV-CPA secure if $\mathcal{A}$ cannot make CCA queries.

Obfuscation. An obfuscator $\mathsf{Obf}(1^\lambda, C)$ is an efficient randomized function which maps circuits to circuits[7]. For correctness, we require that $\mathsf{Obf}(1^\lambda, C)(x) = C(x)$ for all λ, x. We will also consider obfuscators that only work on circuits of a particular format. We now discuss two notions of security.

Definition 2.11 (VGB [BCKP14])**.** Obf is VGB secure if, for all polynomial-time $\mathcal{A}$, all polynomials s, and all inverse polynomials p, there exists a simulator S that is computationally unbounded but which can only make a polynomial number of queries, such that for all circuits C of size at most $s(\lambda)$,
$|\Pr[1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{Obf}(1^\lambda, C))] - \Pr[1 \leftarrow S^C(1^\lambda)]| < p(\lambda)$.

VGB obfuscation is not known under standard assumptions, but it appears plausible that many existing iO constructions satisfy it. Regardless, ruling out VGB obfuscation appears challenging. As we only use VGB for an impossibility, it is still meaningful even if none of the existing candidates are secure. A weakening of VGB obfuscation is *indistinguishability* obfuscation (iO), which is identical except that S can also be query unbounded. An equivalent formulation of iO is that the obfuscations of *equivalent* programs are computationally indistinguishable.

Definition 2.12 (CC security [GKW17, WZ17])**.** For a polynomial s, consider the class of binary circuits of the form "Output 1 on input x if and only if $C(x) = y$" where $y \in \{0,1\}^\lambda$ and C has size s. Call this circuit $\mathsf{CC}_{C,y}(x)$. An obfuscator Obf is a compute-and-compare (CC) obfuscator if it is correct for this class of circuits, and satisfies the following security definition: there exists an efficient simulator S such that for all C and all efficient $\mathcal{A}$,

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(\tilde{C}) : \substack{y \leftarrow \{0,1\}^\lambda \\ \tilde{C} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{CC}_{C,y})}\right] - \Pr[1 \leftarrow \mathcal{A}(S(1^\lambda, 1^s))] \right| < \mathsf{negl}(\lambda) \ .$$

Thas is, if y is random, the obfuscated program can be simulated without knowing C or y at all. [GKW17, WZ17] construct CC-secure obfuscation from LWE.

3 The Augmented Random Oracle Model

3.1 The Plain ROM

In the plain ROM, there is a function $O : \{0,1\}^* \to \{0,1\}^\ell$, where the output of O on any input is chosen uniformly at random. All parties can make queries to O. We call this distribution over oracles $\mathcal{O}$ [8].

[7] We can also consider obfuscators for uniform computational models, but we will not need to for this work.

[8] Note that the choice of ℓ is arbitrary: one can obtain an O with ℓ-bit outputs from an O' with 1-bit outputs by setting $O(x)_i = O(x||i)$. One can even obtain O with infinite outputs in this way. Thus, all random oracles are equivalent.

Complexity Metrics. A query x to O has cost $|x|$. The *query complexity* of an algorithm is the total cost of all its queries. The *computational complexity* is the sum of its query complexity and running time. Both the query and computational complexities of an algorithm can be input-specific. Note the cost must increase with input size to yield correct query complexity results for variable-length O.

Secure cryptosystems in the ROM. Specifications remain oracle-free, but now the cryptosystem Π and adversary $\mathcal{A}$ can query O. We denote the interaction $b \leftarrow (\mathcal{A}^O \leftrightarrow G^{\Pi^O})$. $\mathcal{A}$'s advantage is defined as in the standard model, except that the probability is over the choice of $O \leftarrow \mathcal{O}$. Oracle-free specifications means simulators in simulation-based definitions cannot program O, departing from [BR93]. This modeling, however, automatically captures Type 4 failures [Nie02, Pas03].

Definition 3.1. An oracle-aided cryptosystem Π^O securely implements a specification $\mathcal{G}$ in the ROM if, for all $(G, p) \in \mathcal{G}$ and for all oracle-aided adversaries $\mathcal{A}^O$, there is a negligible negl such that $\mathsf{Adv}_{\mathcal{A}^O, G^{\Pi^O}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Transforms in the ROM. Transforms in the ROM use random oracles. Often in the literature, the underlying building block is prevented from making oracle queries; we will make no such restriction. This models the real world, where the building blocks could have themselves been built using hash functions.

Definition 3.2. An oracle-aided transform T between from $\mathcal{G}$ to $\mathcal{H}$ is secure in the ROM if, for all oracle-aided cryptosystems Π^O which securely implement $\mathcal{G}$ in the ROM, $\Gamma^O = T^{O,\Pi^O}$ securely implements $\mathcal{H}$ in the ROM.

3.2 Augmented Random Oracles

In an augmented random oracle, first a function $O \leftarrow \mathcal{O}$ is sampled. Additionally, there is a distribution $\mathcal{M}$ over oracle-aided functions from which $M \leftarrow \mathcal{M}$ is sampled. O and M are sampled independently. Then, parties are provided with the oracles O and M^O; that is, M's own oracle is set to O. Once O, M are sampled, they are deterministic and stateless. Looking ahead, M will provide one or more abstract cryptosystems. M can still model stateful cryptosystems by having the state be an additional input and output. M itself being stateless corresponds to the typical real-world demand that abstract cryptosystem specifications do not change over time. Note that the restriction to deterministic M is without loss of generality, since any random coins can be provided as an additional input.

Query Complexity. We will treat M as outputting both the output, as well as an arbitrary cost for the query, which may or may not depend on in the input-size or complexity of answering the query. The query complexity of an algorithm making queries to M, O will be the total cost of all *direct* queries, excluding those M makes to O.

Complexity preserving. M is *complexity preserving* if the cost it outputs is at least the query complexity of M when answering that query. In this case, the query complexity of an algorithm is lower bounded by the total cost of *all* queries made to O, including those made by M. There is no cost upper bound.

Simulatable. $\mathcal{M}$ is *simulatable* if, for any distinguisher D, there is an *efficient but stateful* oracle-aided algorithm S^O such that D cannot distinguish the oracles (O, M^O) and (O, S^O) except with negligible probability. Note that many oracles are simulatable via lazy sampling, such as random oracles and generic groups.

Secure cryptosystems in the AROM. Specifications themselves still remain oracle-free. Cryptosystems Π are allowed to make queries to O and M, which we denote by Π^{O,M^O}. We denote the interaction $b \leftarrow (\mathcal{A}^{O,M^O} \leftrightarrow G^{\Pi^{O,M^O}})$. The advantage of $\mathcal{A}$ is defined similarly to the standard model, except that the probability is additionally over the choice of $O \leftarrow \mathcal{O}$ and $M \leftarrow \mathcal{M}$.

Definition 3.3. An oracle-aided cryptosystem Π^{O,M^O} securely implements a specification $\mathcal{G}$ in the $\mathcal{M}$-AROM if, for all $(G, p) \in \mathcal{G}$ and for all oracle-aided adversaries $\mathcal{A}^{O,M^O}$, there exists a negligible function negl such that the advantage of $\mathcal{A}^{O,M^O}$ when interacting with $G^{\Pi^{O,M^O}}$ is at most negl.

Looking ahead, when actually designing cryptosystems, we generally do not want Π to make queries to M. This is because M will model non-black-box techniques, which are generally inefficient in practice. We denote such a protocol by Π^O. In this case, we can quantify over all M, giving the unquantified AROM. Here we do make restrictions on M: namely we require M to be complexity preserving and simulatable.

Definition 3.4. An oracle-aided cryptosystem Π^O (making no queries to M) securely implements $\mathcal{G}$ in the AROM (no quantification by $\mathcal{M}$) if it securely implements $\mathcal{G}$ in the $\mathcal{M}$-ROM for all complexity preserving simulatable $\mathcal{M}$.

Transforms in the ROM. Transforms in the (unquantified) AROM make use of O, but not M, for the same reasons as for cryptosystems. But we always allow the *input* cryptosystems to query M. This will model transform failures, which design input systems employing non-black-box techniques.

Definition 3.5. An oracle-aided transform $T^{O,\Pi}$ from $\mathcal{G}$ to $\mathcal{H}$ is secure in the AROM if, for all complexity preserving simulatable $\mathcal{M}$, and all oracle-aided Π^{O,M^O} which securely implement $\mathcal{G}$ in the $\mathcal{M}$-AROM, $\Gamma^{O,M^O} = T^{O,\Pi^{O,M^O}}$ securely implements $\mathcal{H}$ in the $\mathcal{M}$-AROM.

3.3 Some Basic Results

We show that for direct cryptosystems (not transforms), the AROM and ROM are equivalent for single-stage games:

Theorem 3.6. *If all games in $\mathcal{G}$ are single stage, then Π^O securely implements a specification $\mathcal{G}$ in the AROM if and only if it securely implements $\mathcal{G}$ in the plain ROM.*

An immediate corollary of Theorem 3.6 is that most standard-model properties one assumes of hash functions hold for O in the AROM; for example:

Corollary 3.7. *In the AROM, O is one-way, collision resistant, a pseudorandom generator, and anti-lossy.*

Note, however, that Theorem 3.6 does *not* apply to one-wayness against auxiliary input, since that security definition is not single-stage. As we demonstrate in Section 7, anti-lossiness and auxiliary input one-wayness are incompatible in the standard model, and this incompatibility extends to the AROM. As such, O is *not* auxiliary input one-way in the AROM. We now prove Theorem 3.6.

Proof. Setting $\mathcal{M}$ to always outputs 0, we see that AROM security readily implies ROM security. In the other direction, consider any oracle distribution $\mathcal{M}$ and adversary $\mathcal{A}$ in the AROM. We replace $\mathcal{M}$ with S^H, only negligibly affecting the advantage of $\mathcal{A}$. Now we merge S and $\mathcal{A}$ into a single adversary $\mathcal{A}'$ for Π in the plain ROM. $\mathcal{A}'$ is therefore still an adversary, provided the game is single-stage since it must remember the state of S. The complexity of $\mathcal{A}'$ is polynomially larger than the query complexity of $\mathcal{A}$ (since $\mathcal{M}$ is complexity preserving). Therefore, the overall computational complexity of $\mathcal{A}'$ is only polynomially larger than that of $\mathcal{A}$ in the AROM. Its success probability is negligibly close to that of $\mathcal{A}$. □ □

Note that, unlike or cryptosystems, Theorem 3.6 does *not* hold for transforms because there is no way to simulate Π's queries to M.

4 A Case Study: Encrypt-with-Hash

Here, we use the Encrypt-with-Hash (EwH) transform [BBO07] as a case study. We will see how the uninstantiability result of [BFM15] works, how it the uninstantiability is captured by the AROM, and how to circumvent it. Along the way, we will also see how the assumptions necessary to obtain the uninstantiability can be improved, and how this improvement too is captured by the AROM.

4.1 The (Tweaked) EwH Transform

We first give the Encrypt-with-Hash transform.

Construction 4.1 (Tweaked Encrypt-with-Hash [BBO07])**.** Let $\mathsf{\Pi_{PKE}} = (\mathsf{Gen_{PKE}}, \mathsf{Enc_{PKE}}, \mathsf{Dec_{PKE}})$ be a public key encryption scheme. Define $\mathsf{\Pi_{EwH}} = (\mathsf{Gen_{EwH}}^O, \mathsf{Enc_{EwH}}^O, \mathsf{Dec_{EwH}}^O)$, where

- $\mathsf{Gen_{EwH}}(1^\lambda)$: Run $(\mathsf{pk_{PKE}}, \mathsf{sk_{PKE}}) \leftarrow \mathsf{Gen_{PKE}}(1^\lambda)$, sample $s \leftarrow \{0,1\}^\lambda$, and output $(\mathsf{pk_{EwH}} = (\mathsf{pk_{PKE}}, s), \mathsf{sk_{EwH}} = (\mathsf{sk_{PKE}}, \mathsf{pk_{EwH}}))$.
- $\mathsf{Enc_{EwH}}(\mathsf{pk_{EwH}}, m) = \mathsf{Enc_{PKE}}(\mathsf{pk_{PKE}}, m;\ O(s,m))$
- $\mathsf{Dec_{EwH}}(\mathsf{sk_{EwH}}, c) = \mathsf{Dec_{PKE}}(\mathsf{sk_{PKE}}, c)$.

As discussed in Section 1.4, the original EwH transform did not have s, replacing $O(s,m)$ with $O(\mathsf{pk_{PKE}}, m)$. However, such a construction gives rise to a much simpler Type 3 failure. The problem is that the original transform is only secure in the ROM if $\mathsf{\Pi_{PKE}}$ is not allowed to query O; if we model the random oracle model as allowing $\mathsf{\Pi_{PKE}}$ to query O, then the transform is insecure. In order to avoid that failure, we introduce the tweaked EwH transform given in Construction 4.1, which is secure in the ROM, even when $\mathsf{\Pi_{PKE}}$ can query O.

4.2 Uninstantiability of EwH

Now we explain the uninstantiability result of [BFM15], and how it can be readily be captured in the AROM. Let $\mathsf{\Pi'_{PKE}}$ be any public key encryption scheme, G be a pseudorandom generator, and Obf an obfuscator that is iO secure. [BFM15] use $\mathsf{\Pi'_{PKE}}$ to build a new secure public key encryption scheme $\mathsf{\Pi_{PKE}}$, such that when $\mathsf{\Pi_{PKE}}$ is plugged into Construction 4.1, the resulting $\mathsf{\Pi_{EwH}}$ is insecure, thus invalidating the transform in the standard model.

Construction 4.2 (EwH Uninstantiability)**.** Π_{PKE} is constructed as follows:

- $\mathsf{Gen} = \mathsf{Gen}'$
- $\mathsf{Enc}(\mathsf{pk}_{\mathsf{PKE}}, m;\ r = (r_0, r_1)\ )$: Let $y = \mathsf{G}(r_0)$ and run $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}_{\mathsf{PKE}}, m; y)$. Then run $\tilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P_{m,y}; r_1)$ where $P_{m,y}(f)$ takes as input the code f for some function, and checks if $\mathsf{G}(f(m)) = y$; if so $P_{m,y}$ outputs m, otherwise it outputs $\perp$. Finally output $c = (c', \tilde{P})$.
- $\mathsf{Dec}(\mathsf{sk}, c = (c', \tilde{P}))$: run $\mathsf{Dec}'(\mathsf{sk}, c')$.

[BFM15] prove the following, paraphrased into our terminology:

Theorem 4.3. *If Construction 4.1 is applied to Π_{PKE} from Construction 4.2, then the resulting Π_{EwH} is not weakly CPA-PRIV in the standard model, even against 1-sources, regardless of the hash function used to instantiate O.*

We sketch the proof. When Π_{PKE} is plugged into Construction 4.1, the resulting cryptosystem is completely broken when the random oracle O is replaced by any concrete hash function H. An adversary, given $\mathsf{pk}_{\mathsf{EwH}} = (\mathsf{pk}_{\mathsf{PKE}}, s)$ and $(c', \tilde{P}) \leftarrow \mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m) = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}, m;\ H(s, m)\ )$, constructs the code of the function $f(m')$ which outputs r_0 computed from $(r_0, r_1) \leftarrow H(s, m')$. Then it runs $\tilde{P}(f)$. Recall that $\tilde{P}$ is an obfuscation of $P_{m,\mathsf{G}(r_0)}$, and $P_{m,\mathsf{G}(r_0)}(f)$ evaluates $\mathsf{G}(f(m))$, which for our f is exactly $\mathsf{G}(f(m)) = y$, $\tilde{P}(f)$ outputs m.

It remains to show that Π_{PKE} is a secure public key encryption scheme. We briefly sketch the argument. By pseudorandomness of G, the first step is to replace y with uniformly random bits. At this point, since the image of G is sparse, y is outside the image except with negligible probability. In such a case, the function $P_{m,y}$ is actually equivalent to the function that outputs $\perp$ on all inputs. So by the security of the obfuscator (namely indistinguishability obfuscation, iO), $\tilde{P}$ can be replaced by an obfuscation of the program that outputs $\perp$ everywhere. At this point, $\tilde{P}$ contains no information about m or y, so m is hidden by the assumed security of Π'_{PKE}.

4.3 Translation to the AROM

We now explain how the above is readily captured by the AROM. Concretely, we will prove the following:

Theorem 4.4. *For general CPA secure Π_{PKE}, the (tweaked) EwH is* not *weakly CPA-PRIV in the AROM, even when restricting to 1-sources.*

Proof. M will be the combination of three different M's: M_{PKE} which implements a public key encryption scheme (in order to obtain Π'_{PKE}), M_{G} which implements a pseudorandom generator (to obtain G), and M_{Obf}, which implements an obfuscation scheme (in order to obtain iO).

M_{PKE}**.** Here, we model an ideal public key encryption scheme, following [ZZ20]. Let K, E be random injections. We assume the inverse of an injection outputs $\perp$ if evaluated on a point not in the image. M_{PKE} offers three kinds of queries:

- `gen` queries: takes as input a string sk, and returns $\mathsf{pk} = K(\mathsf{sk})$.
- `enc` queries: takes as input pk, m, r, and returns $E(\mathsf{pk}, m, r)$

- `dec` queries: takes as input sk, c. Compute $d = I^{-1}(c)$. If $d \neq \perp$, then parse $d = (\mathsf{pk}, m, r)$. If $\mathsf{pk} = K(\mathsf{sk})$, then return m. Otherwise return $\perp$.

Relative to M_{PKE}, a public key encryption scheme Π'_{PKE} unconditionally exists: Gen' simply sets sk to be its random coins, and computes and outputs $\mathsf{pk} = K(\mathsf{sk})$ by making a `gen` query. Then $\mathsf{Enc}'(\mathsf{pk}, m; r) = E(\mathsf{pk}, m, r)$ using an `enc` query, and $\mathsf{Dec}'(\mathsf{sk}, c)$ makes a `dec` query on sk, c to produce m. As explained by [ZZ20], the resulting scheme is readily shown to be CPA secure (and much more) against query-bounded adversaries.

M_{G}. This is just an *expanding* random oracle. Namely, M_{G} will just be an expanding random oracle G, independent of K, E. Expanding random oracles are trivially pseudorandom generators.

M_{Obf}. This is the obfuscation model proposed by Asharov and Segev [AS15], except extended to allow programs that also query M_{G}. Let I be a random injection. Then M_{Obf} will offer two kinds of queries:

- `obfuscate` queries: takes as input the description of a program P and random coins r; it returns $\tilde{P} = I(P, r)$.
- `eval` queries: takes as input a string $\tilde{P}$ and input x. Compute $d \leftarrow I^{-1}(\tilde{P})$. If $d \neq \perp$, then parse d as P, r and output $P(x)$. Otherwise output $\perp$.

Importantly, we will allow the inputs P to `obfuscate` to be *oracle-aided* programs, making queries to G and more importantly to O. During the computation of $P(x)$ in an `eval` query, M_{Obf} will forward queries to G to M_{G} and queries to O to the random oracle O. We can then have $\mathsf{Obf}(P; r)$ simply make an `obfuscate` query on (P, r). It is straightforward that Obf is iO secure for such oracle-aided programs, and even VGB secure. In fact, it is even virtual *black* box (VBB) secure, which is known to be impossible in the standard model. However, we only need that it is iO. In fact, we could replace M_{Obf} with any model that implements obfuscation, so long as (1) the obfuscator was iO secure, and (2) the input programs could query G and O.

The final M is just the combination $(M_{\mathsf{PKE}}, M_{\mathsf{G}}, M_{\mathsf{Obf}})$. At this point, the proof of Theorem 4.4 proceeds almost identically to Section 4.2, with Π'_{PKE} instantiated by M_{PKE}, G instantiated by M_{G}, and Obf instantiated by M_{Obf}. The main difference is that we need f to be oracle-aided, making queries to O. In turn, this means $P_{m,y}$ must also be oracle-aided, now making queries to both O and G. Fortunately, M_{Obf} acts on such oracle-aided programs. We derive Π_{PKE} as in Construction 4.2, and security of Π_{PKE}, relative to M, follows by an identical argument relying on the security of $\Pi'_{\mathsf{PKE}}, \mathsf{G}$, and Obf.

The attack is also quite similar. An adversary, given $\mathsf{pk}_{\mathsf{EwH}} = (\mathsf{pk}_{\mathsf{PKE}}, s)$ and $(c', \tilde{P}) \leftarrow \mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m) = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}, m;\ O(s, m)\ )$, constructs the (oracle-aided) code of the function $f(m')$ with s hardcoded, which outputs r_0 computed from $(r_0, r_1) \leftarrow O(s, m')$. This f makes queries to O.

Then it computes $\tilde{P}(f)$ by making an `eval` query on (P, f). M_{Obf} will respond by computing $P_{m,y}(f)$ where $y = G(r_0)$. $P_{m,y}(f)$ evaluates $G(f(m))$, which for our f is exactly y. Since $G(f(m)) = y$, the `eval` query outputs m in the clear. This completes the proof of Theorem 4.4. □

4.4 An Improved Uninstantiability

Here, we improve the computational assumptions needed for the uninstantiability of EwH. While a potentially interesting fact on its own, our improvement also illustrates the need for flexibility in M in order for the AROM to capture a wide variety of uninstantiability results.

Concretely, we show that it suffices to assume compute-and-compare obfuscation and fully homomorphic encryption, both of which can be instantiated under circularly-secure LWE.

Construction 4.5 (Impossibility)**.** Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any public key encryption scheme, $\Pi_{\mathsf{fhe}} = (\mathsf{Gen}_{\mathsf{fhe}}, \mathsf{Enc}_{\mathsf{fhe}}, \mathsf{Dec}_{\mathsf{fhe}}, \mathsf{Eval}_{\mathsf{fhe}})$ be any FHE scheme, and Obf a compute-and-compare obfuscator. Let G be a PRG. Define $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ to be the following

- $\mathsf{Gen}_{\mathsf{PKE}}(1^\lambda) = \mathsf{Gen}(1^\lambda)$.
- $\mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}, m)$: choose a random r and compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$. Sample $(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{fhe}}) \leftarrow \mathsf{Gen}_{\mathsf{fhe}}(1^\lambda)$ and compute $d \leftarrow \mathsf{Enc}_{\mathsf{fhe}}(\mathsf{pk}_{\mathsf{fhe}}, m)$. Let $y_0 = G(r)$ and y_1 be uniformly random. Let b the first bit of m. Finally let $\tilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{CC}_{G(\mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}_{\mathsf{fhe}}, \cdot)), y_b})$ [9]. Output $(c, \mathsf{pk}_{\mathsf{fhe}}, d, \tilde{P})$.
- $\mathsf{Dec}_{\mathsf{PKE}}(\mathsf{sk}, (c, \mathsf{pk}_{\mathsf{fhe}}, d, \tilde{P}))$: Output $\mathsf{Dec}(\mathsf{sk}, c)$.

The following theorems prove the uninstantiability result.

Theorem 4.6. *If Construction 4.1 is applied to Π_{PKE} from Construction 4.5, then the resulting Π_{EwH} is not weakly CPA-PRIV in the standard model, even against 1-sources, regardless of the hash function used to instantiate O.*

Proof. Let D be the uniform distribution over all messages whose first bit is 1, with no auxiliary input. This is clearly statistically unpredictable. Consider a ciphertext $(c, \mathsf{pk}_{\mathsf{fhe}}, d, \tilde{P})$ generated as $\mathsf{Enc}_{\mathsf{EwH}}((\mathsf{pk}, s), m) = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}, m; H(\mathsf{hk}, s, m))$, where $H(\mathsf{hk}, \cdot, \cdot)$ is the concrete hash function implementing the random oracle. Let $H'(\mathsf{hk}, s, m)$ denote the truncation of H to the randomness used to construct c.

Compute $c' \leftarrow \mathsf{Eval}_{\mathsf{fhe}}(\mathsf{pk}_{\mathsf{fhe}}, d, H'(\mathsf{hk}, s, \cdot))$, where c' encrypts $H'(\mathsf{hk}, s, m) = r$, the randomness used to generate c. Then compute $\tilde{P}(c')$. Observe that $\mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}_{\mathsf{fhe}}, c') = r$, and so $\tilde{P}(c')$ outputs 1 if and only if $y_b = G(r)$. Thus, $\tilde{P}(c')$ outputs the first bit of m. This violates weak CPA-PRIV security. □

Theorem 4.7. *If Π is sub-exponentially CPA secure, $\Pi_{\mathsf{fhe}}, \mathsf{Obf}$ are polynomially secure, and G is pseudorandom against sub-exponentially hard computationally unpredictable sources, then Π_{PKE} is CPA secure.*

Note that the necessary G can be constructed from sub-exponentially hard LWE, following ideas from Brakerski and Segev [BS11] and Zhandry [Zha16].

Proof. To prove security, we just need to show that $\tilde{P}$ can be replaced with a simulated program that does not depend on r or $\mathsf{sk}_{\mathsf{fhe}}$; then we can rely on the security of Π and Π_{fhe} to hide m. In the case $b = 1$, this is trivial: y_b is a uniformly random string, so we can invoke compute and compare security to see that we can simulate $\tilde{P}$ as $\tilde{P} \leftarrow S(1^\lambda, 1^{|P|})$. To handle the case $b = 0$, we show that the cases where $\tilde{P}$ is generated using y_0 and y_1 are indistinguishable, and then rely on the $b = 1$

[9] Recall $\mathsf{CC}_{G(\mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}_{\mathsf{fhe}}, \cdot)), y_b}$ is the program $x \mapsto G(\mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}_{\mathsf{fhe}}, x)) == y_b$.

case to simulate. y_0 and y_1 are indistinguishable since r is sub-exponentially hard to predict (by the sub-exponential hardness of Π) and the fact that G is pseudorandom against sub-exponentially hard sources. □

We note that Fujisaki-Okamoto [FO99] (FO, defined in Section 5) bears many similarities to EwH. As with the original EwH impossibility of [BFM15], essentially the same impossibility applies to FO. We omit the details.

4.5 Other Possible Oracles

Our improved uninstantiability result shows that it is also important to consider oracles other than M_{Obf}, to adequately capture all the non-black-box techniques that may be used. It is not difficult to come up with oracles M_{FHE} that implement fully homomorphic encryption, where the homomorphic operations may include O gates. This allows the AROM to capture our improved uninstantiability above. Another limitation of M_{Obf} pointed out by Asharov and Segev [AS15] is that it fails to capture NIZKs, another common tool in constructions using iO. In the AROM, we can easily create an oracle M_{NIZK} that provides a NIZK proof functionality for statements that involve queries to O. We can similarly define oracles for any other non-black box tool applied to circuits that involve O queries, and even oracles combining all of the above. Thus, as long as the non-black-box techniques are simply using that the hash function has code that can be run—but not using any particular features of that code—it seems that all such techniques are captured by the AROM. Hence, AROM security provides compelling resiliency to such techniques. This will be the focus of Section 4.6.

Non-examples. There are oracles that are non-examples. Most prominently would be Simon's oracle [Sim98], which finds collisions in functions without violating one-wayness. This oracle makes exponentially-many queries to the random oracle, thereby learning its entire truth table, and also cannot be efficiently simulated in any way. More generally, Simon's oracle is an example of the common "two oracle trick" in black box separations, where one oracle implements a cryptosystem $\mathcal{B}$, but another oracle is designed to break any instantiation of $\mathcal{C}$.

4.6 Overcoming ROM Failures for EwH

We now explain how to achieve deterministic encryption in the AROM, despite known uninstantiability results for EwH working in the AROM.

At first glance, proving security in the AROM appears non-trivial: how do you reason about *any* possible oracle M, which may implement arbitrarily complex functionalities? Imagine, for example, that M directly provides oracles implementing a cryptosystem Π as in M_{PKE}. But M knows the injections being used to define the cryptosystem, meaning M itself can internally invert this injection, learning the secret key for any public key. For a given transform T, one could plausibly augment this M with a `break` functionality, that breaks Π whenever it is used inside T, but leaves Π as secure when used "honestly." Any security proof would have to rule out such a M.

Now, the obvious solution is a reduction: showing that any adversary for T^{Π} can be converted into an attacker for Π. Thus, if M provided a mechanism to break T^{Π}, it would contradict that Π is secure. This is the approach we follow. But in the AROM, devising a reduction is nevertheless non-trivial. If the reduction could be performed exactly as it would in the standard model, there is no need to work in the ROM or AROM in the first place. So for the transforms we consider,

the reduction will be making non-trivial use of the random oracle O. But it also cannot just freely program the random oracle seen by the adversary as in typical ROM proofs: the reduction must result in an adversary for Π, which makes queries to M (since that is how Π was constructed) which in turn makes queries to O. Thus if we try re-programming O, it will be inconsistent with the "true" O. Since the adversary has indirect access to the true O, this re-programming could plausibly be detected, causing the adversary to abort. How do we structure the proof in such a way that this detection is not possible, *regardless* of the structure of M?

Looking ahead, our solution will first use the security property of Π to move to a hybrid; this step is a standard reduction and does not make any particular use of the random oracle. Then in the hybrid, a *statistical* property will hold. This statistical property allows us to establish some security against M itself, which we then use to carefully program the random oracle, etc.

We now explore how to implement this vague idea in the case of EwH (Construction 4.1) in order to make it secure. In particular, we consider strengthening Π to being *lossy* (Definition 2.8). Observe that the uninstantiability of [BFM15] detailed in Section 4.2 uses a non-lossy public key encryption scheme. Afer all, the program $\tilde{P}$ is an obfuscation of $P_{m,y}$, which has the message m hard-coded. While m is presumably hidden computationally, m determines the program's behavior and therefore is information-theoretically determined by $\tilde{P}$. Thus, even if, say, the original encryption scheme Π'_{PKE} were lossy, the resulting scheme in Construction 4.2 will never be lossy.

We show that this limitation of [BFM15] is inherent: that EwH is weakly CPA-PRIV in the AROM if using a lossy encryption scheme[10]. Since the techniques of [BFM15] are captured by the AROM, we thus show that the techniques cannot extend to EwH when using lossy encryption. We note that weak CPA-PRIV implies correlated-input secure one-way functions, which Wichs [Wic13] shows cannot be proved secure using black-box reductions to any falsifiable assumption. This means some idealized model is necessary for security of EwH.

Theorem 4.8. *If Π_{PKE} is lossy, then Π_{EwH} is weakly CPA-PRIV in the AROM.*

Proof. Consider a distribution $\mathcal{M}$ over oracles M, and some lossy encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}{}^{O,M^O}, \mathsf{Enc}_{\mathsf{PKE}}{}^{O,M^O}, \mathsf{Dec}_{\mathsf{PKE}}{}^{O,M^O}, \mathsf{GenLossy}_{\mathsf{PKE}}{}^{O,M^O})$ in the $\mathcal{M}$-AROM. Let $\Pi_{\mathsf{EwH}}{}^{O,M^O}$ be the result of applying EwH to Π_{PKE}.

Consider an ℓ-source D^{O,M^O}, which we assume to be statistically unpredictable in the $\mathcal{M}$-AROM. Let $\mathcal{A}^{O,M^O}$ be a CPA-PRIV adversary with advantage ϵ. We now define hybrids:

Hybrid 0. Here, $\mathcal{A}$ plays the CPA-PRIV game against Γ and D, with $b = 0$. This means $\mathcal{A}$ receives aux and encryptions of $m^*_{i,0}$, where $(m^*_{1,0}, \ldots, m^*_{\ell,0}, \mathsf{aux}) \leftarrow D^{O,M^O}(1^\lambda)$. Let p_0 be the probability $\mathcal{A}$ outputs 1.

Hybrid 1. Here, we switch $\mathsf{pk}_{\mathsf{PKE}}$ to be generated by $\mathsf{pk}_{\mathsf{PKE}} \leftarrow \mathsf{GenLossy}_{\mathsf{PKE}}(1^\lambda)$. Let p_1 be the probability $\mathcal{A}$ outputs 1. By the assumed lossiness of Π_{PKE} in the $\mathcal{M}$-AROM, we must have $|p_1 - p_0|$ is negligible.

Hybrid 2. This is identical to Hybrid 1, except that the experiment immediately aborts if $\mathsf{GenLossy}$ or D (when being run by the experiment) ever make a query $O(s, m^*_{i,b})$ for some $i \in [\ell], b \in \{0,1\}$, or if they make a query to M that triggers such a query to O. Here, $m^*_{i,1}$ are sampled as uniformly

[10] We do not know how to prove CCA-PRIV or strong security for this construction.

random distinct messages. Let p_2 be the probability $\mathcal{A}$ outputs 1. Notice that $\mathsf{GenLossy}$ and D only receive 1^λ as input, and so are independent of s. Since $\mathcal{M}$ is complexity preserving and $\mathsf{GenLossy}, D$ are efficient, they can only trigger a polynomial number of queries each, so the probability of such a query is negligible. Hence $|p_2 - p_1|$ is negligible.

Observe that in Hybrid 2, the very first queries to $O(s, m^*_{i,0})$ for any $i \in [\ell]$ are when running $\mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m^*_{i,b}) = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}, m^*_{i,0};\ O(s, m^*_{i,0})\ )$. Note that each of the $m^*_{i,0}$ are distinct, so all such first queries are distinct.

Hybrid 3. This is the same as Hybrid 2, except that the experiment aborts if there are any queries to $O(s, m^*_{i,b})$ occurring *after* those by $\mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m^*_{i,0})$. Let p_3 be the probability $\mathcal{A}$ outputs 1. We will prove $|p_3 - p_2|$ is negligible shortly.

In Hybrid 3, since $O(s, m^*_{i,0})$ is only ever answered once, namely inside $\mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m^*_{i,0})$, the random coins generated in each call to $\mathsf{Enc}_{\mathsf{EwH}}$ are random and independent of the rest of the experiment. Thus, Hybrid 3 is equivalent to giving $\mathcal{A}$ the ciphertexts $\mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}, m^*_{i,0})$ for fresh random coins.

Hybrid 4. This is the same as Hybrid 3, except that we switch the ciphertexts given to $\mathcal{A}$ to be $\mathsf{Enc}_{\mathsf{EwH}}(\mathsf{pk}_{\mathsf{EwH}}, m^*_{i,1})$ for uniformly random $m^*_{i,1}$. Let p_4 be the probability $\mathcal{A}$ outputs 1. As in Hybrid 3, the experiment is equivalent to the ciphertexts being $\mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}, m^*_{i,1})$ for fresh random coins. By the lossiness of $\mathsf{Enc}_{\mathsf{PKE}}$, $|p_4 - p_3|$ is negligible.

We now prove $|p_3 - p_2|$ negligible. First, since $m^*_{i,1}$ are random distinct messages that are independent of the view of the experiment in Hybrid 2, the probability of querying on $O(s, m^*_{i,1})$ is negligible. Now we consider the first query of the form $O(s, m^*_{i,0})$ triggered after $\mathsf{Enc}_{\mathsf{EwH}}$ in Hybrid 2. Up until this point, Hybrids 2, 3, and 4 is statistically close. But in Hybrid 4, prior to any $O(s, m^*_{i,0})$ query, the experiment only uses $m^*_{i,1}$, and not the $m^*_{i,0}$. Hence, the $m^*_{i,0}$ remains statistically independent of the view of the experiment up until this point. Making a query on $m^*_{i,0}$ would thus violate the statistical unpredictability of D, and hence can only occur with negligible probability.

Hybrids 5,6, and 7. These are the same as Hybrids 2,1, and 0, respectively, except that the messages being encrypted are $m^*_{i,1}$. Let p_5, p_6, p_7 be the probabilities of outputting 1. By analogous arguments, we have that $|p_5 - p_4|, |p_6 - p_5|, |p_7 - p_6|$ are all negligible. Hence $|p_7 - p_0|$ is negligible. But notice that Hybrid 7 is exactly the CPA-PRIV game with $b = 1$, and so $|p_7 - p_0| = \epsilon$ is the advantage of $\mathcal{A}$. This completes the proof. □

5 Fujisaki-Okamoto in the AROM

Here, we explore the insecurity of the Fujisaki-Okamoto (FO) transform [FO99] in the AROM. Recall that FO starts with $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ and $\Pi_{\mathsf{SKE}} = (\mathsf{Enc}_{\mathsf{SKE}}, \mathsf{Dec}_{\mathsf{SKE}})$, which are public key and secret key encryption schemes. Ciphertexts are then

$$(\ c := \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}, \delta;\ O(0, \delta, d)\ )\ ,\ d := \mathsf{Enc}_{\mathsf{SKE}}(\ O(1, \delta)\ , m)\ )\quad .$$

Note that, because $\mathsf{Enc}_{\mathsf{PKE}}$ never "sees" d, the Type 2 impossibility of the un-tweaked EwH does not seem to apply. For simplicity, we therefore stick with the usual description of FO; we could also define a tweaked version with an s as in Section 4.1, and everything we say below will still apply.

That FO is insecure for general PKE already follows from [BFM15] following a similar proof as the EwH setting, and the insecurity readily carries over to the AROM following a very similar outline as in Section 4.3. In fact, unlike EwH, FO remains insecure in the AROM, even if Π_{PKE} is lossy:

Theorem 5.1. *For general* lossy Π_{PKE} *and even* perfectly secure Π_{SKE}*, FO is* not *secure in the AROM.*

Proof. We start with an oracle M^O which contains families of private random permutations P, Q, and answers the following queries:

- $(\texttt{Gen}, 1^\lambda, s)$: Output $(\mathsf{pk} = P(s,0), \mathsf{sk} = s)$.
- $(\texttt{GenLossy}, 1^\lambda, s)$: Output $\mathsf{pk} = P(s,1)$.
- $(\texttt{Enc}, \mathsf{pk}, m, r)$: If $P^{-1}(\mathsf{pk}) = (\mathsf{sk}, 0)$ for some sk, output $c = Q(\mathsf{pk}, m, r)$. Otherwise output $c = Q(\mathsf{pk}, 0, r)$.
- $(\texttt{Dec}, \mathsf{sk}, c)$: Compute $(\mathsf{pk}, m, r) = Q^{-1}(c)$. If $\mathsf{pk} = P(\mathsf{sk}, 0)$, output m. Otherwise output $\perp$.
- $(\texttt{Forward}, x)$: Output $O(x)$.

M clearly can be used to realize a lossy encryption scheme Π_{PKE}. We instantiate $\mathsf{Enc}_{\mathsf{SKE}}$ with the one-time pad. Let $\Pi_{\mathsf{FO}} = ({\mathsf{Gen}_{\mathsf{FO}}}^{O,M^O}, {\mathsf{Enc}_{\mathsf{FO}}}^{O,M^O}, {\mathsf{Dec}_{\mathsf{FO}}}^{O,M^O})$ be the result of applying the FO transformation to this lossy encryption scheme. Under M as is, Π_{FO} actually will be CCA-secure. We now add two more types of queries to M, which make use of another private random oracle R.

- $(\texttt{EncRand}, \mathsf{pk})$: Compute $(m, r) = R(\mathsf{pk})$ and output $c \leftarrow {\mathsf{Enc}_{\mathsf{FO}}}^{O,M^O}(\mathsf{pk}, m; r)$
- $(\texttt{Break}, \mathsf{pk}, m)$: Compute $(m', r) = R(\mathsf{pk})$ and $(\mathsf{sk}, b) \leftarrow P^{-1}(\mathsf{pk})$. If $m = m'$, output sk.

We claim that the addition of these queries preserves the lossiness of Π_{PKE}. Indeed, suppose an adversary is trying to distinguish pk being lossy from regular. An `EncRand` query on pk does not help: it is just an encryption of a random ciphertext under FO, which the adversary could simulate for itself. On the other hand, suppose it makes a `Break` query on (pk, m) that causes it to output sk. Consider the first such query. In this case, the adversary must have been able to previously learn the plaintext encrypted in the `EncRand` query. Since the query was just a random ciphertext, such an adversary can be turned into an adversary against the CPA-security for Π_{FO} in the setting of only `Gen`, `GenLossy`, `Enc`, `Dec` queries, which we already know is impossible.

However, these queries clearly allow for for CCA attacks on Π_{FO}: simply make an `EncRand` query on the public key, and then make a CCA query on the resulting ciphertext. Then feed the result into a `Break` query, revealing the secret key. □

The above "attack" is quite general: it is not clear that it used any particular structure of Π_{FO}. In the following subsection, we will nevertheless show how to modify the construction to achieve CCA security. Very roughly, the way we get around the issue above is by having a public key comprise of several public keys for Π_{PKE}. What we will see is that this lets us simulate CCA queries by ourselves. Then the ability to perform `EncRand` and `Break` queries will directly allow us to break the security of the underlying encryption scheme. Note that our proof will be much more general, applying to any oracle M.

5.1 Our CCA-secure Construction

Construction 5.2 (CCA-Secure PKE in the AROM)**.** Let $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ and $\Pi_{\mathsf{SKE}} = (\mathsf{Enc}_{\mathsf{SKE}}, \mathsf{Dec}_{\mathsf{SKE}})$ be public key and secret key encryption schemes, respectively. Let $\Pi_{\mathsf{Sig}} = (\mathsf{Gen}_{\mathsf{Sig}}, \mathsf{Sign}_{\mathsf{Sig}}, \mathsf{Ver}_{\mathsf{Sig}})$ be a signature scheme. Define $\Pi_{\mathsf{CCA}} = ({\mathsf{Gen}_{\mathsf{CCA}}}^{O}, {\mathsf{Enc}_{\mathsf{CCA}}}^{O}, {\mathsf{Dec}_{\mathsf{CCA}}}^{O})$, where

- ${\mathsf{Gen}_{\mathsf{CCA}}}^{O}(1^\lambda)$: Let ℓ be the bit-length of vk generated by $\mathsf{Gen}_{\mathsf{Sig}}(1^\lambda)$. For $i \in [\ell], b \in \{0,1\}$, run $({\mathsf{pk}_{\mathsf{PKE}}}^{(i,b)}, {\mathsf{sk}_{\mathsf{PKE}}}^{(i,b)}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$. Output $\mathsf{pk}_{\mathsf{CCA}} = ({\mathsf{pk}_{\mathsf{PKE}}}^{(i,b)})_{i,b}$ and $\mathsf{sk}_{\mathsf{CCA}} = (({\mathsf{sk}_{\mathsf{PKE}}}^{(i,b)})_{i,b}, \mathsf{pk}_{\mathsf{CCA}})$.
- ${\mathsf{Enc}_{\mathsf{CCA}}}^{O}(\mathsf{pk}_{\mathsf{CCA}}, m)$: Sample $(\mathsf{vk}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Gen}_{\mathsf{Sig}}(1^\lambda)$. Sample $\delta \leftarrow \{0,1\}^\lambda$. Run $d \leftarrow \mathsf{Enc}_{\mathsf{SKE}}(O(\mathsf{vk}, \delta)\ ,\ m)$, $c_i \leftarrow \mathsf{Enc}_{\mathsf{PKE}}({\mathsf{pk}_{\mathsf{PKE}}}^{(i,\mathsf{vk}_i)}, \delta;\ O(\delta, i, d, \mathsf{vk})\ )$ for $i \in [\ell]$. Finally compute $\sigma \leftarrow \mathsf{Sign}_{\mathsf{Sig}}(\mathsf{sk}_{\mathsf{Sig}},\ ((c_i)_i, d)\ )$. Output $c = (\mathsf{vk}, (c_i)_i, d, \sigma)$.
- ${\mathsf{Dec}_{\mathsf{CCA}}}^{O}(\mathsf{sk}_{\mathsf{CCA}}, c)$: First run $\mathsf{Ver}_{\mathsf{Sig}}(\mathsf{vk},\ ((c_i)_i, d)\ , \sigma)$; if it rejects immediately abort and output $\perp$. Otherwise run $\delta \leftarrow \mathsf{Dec}_{\mathsf{PKE}}({\mathsf{sk}_{\mathsf{PKE}}}^{(1,\mathsf{vk}_1)}, c_1)$. For each $i > 1$, check that $c_i = \mathsf{Enc}_{\mathsf{PKE}}({\mathsf{pk}_{\mathsf{PKE}}}^{(i,\mathsf{vk}_i)}, \delta;\ O(\delta, i, d, \mathsf{vk})\ )$; if any of the checks fail immediately abort and output $\perp$.

  Finally, output $m \leftarrow \mathsf{Dec}_{\mathsf{SKE}}(O(\mathsf{vk}, \delta)\ ,\ d)$.

Correctness is immediate from the correctness of the underlying protocols. We now state the security theorem:

Theorem 5.3. *If Π_{PKE} is lossy, Π_{SKE} is one-time secure, and Π_{Sig} is strongly one-time secure, then Π_{CCA} is CCA secure in the AROM.*

Proof. Consider a distribution $\mathcal{M}$ over oracles M and schemes $\Pi_{\mathsf{PKE}}, \Pi_{\mathsf{SKE}}, \Pi_{\mathsf{SKE}}$ that are secure in the $\mathcal{M}$-AROM. Let $\mathcal{A}^{O, M^O}$ be a CCA adversary with advantage ϵ. We prove security though a sequence of hybrids:

Hybrid 0. This is the CCA experiment with $b = 0$, meaning m_0^* is encrypted. Let p_0 be the probability the adversary outputs 1. Let $(\mathsf{vk}^*, {\mathsf{sk}_{\mathsf{Sig}}}^*)$ be the signature keys generated for the challenge ciphertext. Note that we can sample these at the beginning of the experiment.

Hybrid 1. This is identical to Hybrid 0, except that we respond with $\perp$ to any CCA query $(\mathsf{vk}, (c_i)_i, d, \sigma)$ such that $\mathsf{vk} = \mathsf{vk}^*$. Let p_1 be the probability $\mathcal{A}$ outputs 1.

The only difference between Hybrid 0 and Hybrid 1 occurs if there is a CCA query where σ is a valid signature on $((c_i)_i, d)$ relative to vk^*. Since CCA queries must be distinct from the challenge query, this immediately yields a signature forgery. By straightforward reduction from the strong 1-time security of Π_{Sig}, we have that $|p_1 - p_0|$ is negligible.

Hybrid 2. This is identical to Hybrid 1, except that in answering CCA queries, we replace $\delta \leftarrow \mathsf{Dec}_{\mathsf{PKE}}({\mathsf{sk}_{\mathsf{PKE}}}^{(1,\mathsf{vk}_1)}, c_1)$ with $\delta \leftarrow \mathsf{Dec}_{\mathsf{PKE}}({\mathsf{sk}_{\mathsf{PKE}}}^{(j,\mathsf{vk}_j)}, c_1)$, where j is the first index such that $\mathsf{vk}_j \neq \mathsf{vk}_j^*$. Since $\mathsf{vk} \neq \mathsf{vk}^*$ such an index must exist. Since $\mathsf{Dec}_{\mathsf{CCA}}$ always checks that $c_i = \mathsf{Enc}_{\mathsf{PKE}}({\mathsf{pk}_{\mathsf{PKE}}}^{(i,\mathsf{vk}_i)}, \delta;\ O(\delta, i, d, \mathsf{vk})\ )$ for all i, the correctness of Π_{PKE} implies that there is negligible probability this change will affect the output of the CCA query. Therefore, if p_2 is the probability $\mathcal{A}$ outputs 1, we have $|p_2 - p_1|$ is negligible. Note that, in Hybrid 2, we no longer need ${\mathsf{sk}_{\mathsf{PKE}}}^{(i,\mathsf{vk}_i^*)}$ for any i.

Hybrid 3. This is identical to Hybrid 2, except that we change $\mathsf{pk}_{\mathsf{PKE}}^{(i,\mathsf{vk}_i^*)} \leftarrow \mathsf{GenLossy}_{\mathsf{PKE}}(1^\lambda)$. If p_3 is the probability $\mathcal{A}$ outputs 1, we have $|p_3 - p_2|$ is negligible, by straightforward reduction to the lossiness of Π_{PKE}.

Hybrid 4. The challenge ciphertext contains $d^* \leftarrow \mathsf{Enc}_{\mathsf{SKE}}(O(\mathsf{vk}^*, \delta^*), m_0^*)$ and the various $c_i^* \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}^{(i,\mathsf{vk}_i^*)}, \delta;\ O(\delta^*, i, d^*, \mathsf{vk}^*)\ )$. In Hybrid 4, we replace these with $d^* \leftarrow \mathsf{Enc}_{\mathsf{SKE}}(k^*, m_0^*)$ for a uniform key k^* and $c_i^* \leftarrow \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_{\mathsf{PKE}}^{(i,\mathsf{vk}_i^*)}, \delta)$ using fresh randomness for each ciphertext.

That $|p_4 - p_3|$ is negligible follows from analogous arguments to the proof of Theorem 4.8: Hybrid 3 and 4 are identical until there is a query to $O(\delta^*, i, d^*, \mathsf{vk}^*)$ or $O(\mathsf{vk}^*, \delta^*)$ occurring outside the generation of the d, c_i^*. But in Hybrid 4, by the lossiness of $\mathsf{pk}_{\mathsf{PKE}}^{(i,\mathsf{vk}_i^*)}$, the entire experiment is independent of the random δ^* up until such a query. So the probability of querying O on any of these queries is therefore negligible.

Hybrid 5. Here we replace $d^* \leftarrow \mathsf{Enc}_{\mathsf{SKE}}(k^*, m_0^*)$ with $d^* \leftarrow \mathsf{Enc}_{\mathsf{SKE}}(k^*, m_1^*)$. If we let p_5 be the probability $\mathcal{A}$ outputs 1, then $|p_5 - p_4|$ is negligible by the one-time security of Π_{SKE}.

Hybrids 6,7,8,9. These are identical to Hybrids 3,2,1, and 0, except that m_0^* is replaced by m_1^*. Let p_6, p_7, p_8, p_9 be the probabilities of outputting 1. Analogous arguments show that $|p_6 - p_5|, |p_7 - p_6|, |p_8 - p_7|, |p_9 - p_8|$ are negligible. Hence $|p_9 - p_0|$ is negligible. But notice that Hybrid 9 is exactly the CCA game with $b = 1$, and so $|p_9 - p_0| = \epsilon$ is the advantage of $\mathcal{A}$. This completes the proof. □

6 Fiat-Shamir in the AROM

Fiat-Shamir (FS) [FS87] reduces interactive public coin protocols into a single message. There are two variants: interactive into non-interactive arguments, and identification protocols into signatures. We will focus on argument systems.

Let $P(x, w) \leftrightarrow V(x)$ be a proof system for an NP language L, where x is an instance and w a witness. The system is a sound proof if, for any $x \notin L$ and any potentially unbounded cheating prover $P^*(x)$, $P^*(x) \leftrightarrow V(x)$ causes V to accept with only negligible probability. The system is a sound *argument* if the above holds for only computationally efficient P^*. The system is *public coin* if V's messages are uniform random strings.

Consider a 3-message public coin proof system, where the prover goes first. Let (a, c, r) be the three messages. The Fiat-Shamir transform compiles such a system into a non-interactive proof, by running $P(x, w)$ but where the verifier's message c is set to $O(s, a)$, resulting in $(a, c = O(s, a), r)$. Here, s is a common reference string (CRS), and is needed to enforce domain separation to avoid trivial Type 2 impossibilities if the underlying proof system can query O. The verifier then just checks the validity of the transcript (a, c, r), and also that $c = O(s, a)$.

Heuristically, one may expect the resulting system to be sound, since the soundness of P, V relied on the un-predictability of c, which seems to hold when deriving c from a good hash function. In the ROM, one can prove this intuition, as shown by [BR93]. Unfortunately, Goldwasser and Kalai [GK03] show that this is not the case in the standard model for general arguments. We show that the insecurity also extends to the AROM:

Theorem 6.1. *For general arguments, FS is* not *secure in the AROM.*

Proof. We could design an M to instantiate the tools used by [GK03], and translate their impossibility to the AROM. However, in the AROM there is a much more direct impossibility. Fix some language L. Let M^O be the oracle with private random permutation P and random functions Q, R, which accepts the following queries:

- `Prove1`: These queries represent the prover's first message. They take as input an instance x and random coins t. It outputs $a = P(x, t)$.
- `Prove2`: These queries represent the prover's second message. They take as input x, t, the also the verifiers challenge c, and also a witness w. If checks if w is a valid witness for x, and returns $\perp$ if not. Assuming w is a valid witness, it outputs $r = Q(x, t, c)$.
- `Ver`: These queries represent the verifier's procedure. They take as input a, c, r. Compute $(x', t) = P^{-1}(a)$, and check that $r = Q(x, t, c)$ and also that $x = x'$. If so, accept; otherwise reject.
- `Attack`: These queries take as input the description f for an oracle-aided function, an instance x. Compute $t = R(x, f)$. Then output $(a = P(x, t), c = f^O(a), r = Q(x, t, c))$.

`Prove1`, `Prove2`, `Ver` give a 3-message public coin argument system. We need to prove that the argument system is sound, and also that applying Fiat-Shamir results in an insecure protocol. We start with the insecurity of Fiat-Shamir. Given x, s, the attacker lets $f^O(a) = O(s, a)$. Then my making an `Attack` query on f, x, the attacker gets (a, c, r) which are accepted by `Ver`.

We now justify the security of `Prove1`, `Prove2`, `Ver`. Let x be a false instance. Consider a supposed adversary for the 3-round protocol. Let the three messages be a^*, c^*, r^*. Note that once a^*, c^* are chosen, there is at most a single r^* that will cause the verifier to accept, namely $r^* = Q(x, t^*, c^*)$ where $(x, t^*) = P^{-1}(a^*)$. Since Q is a random function, the value of r^* is perfectly independent of the adversary's view until some query the adversary makes triggers a query query $Q(x, t^*, c^*)$.

Queries to `Prove1` do not trigger any query to Q. Note that since x is false, no query to `Prove2` can trigger a query $Q(x, t^*, c^*)$. A query to `Attack` will query $Q(x, t, c)$ where t is a random value derived as $t = R(x, f)$. We consider two cases:

- The query to `Attack` was made before seeing c^*. In this case, c^* is independent of the query, so with overwhelming probability, $c^* \neq c$. Therefore, no query $Q(x, t^*, c^*)$ will be made.
- The query to `Attack` was made after seeing c^*. In this case, the adversary already committed to a^* and hence t^*. As such, the value of t was independent of the adversary's view when it committed to t^*, and so $t \neq t^*$ with overwhelming probability. Therefore, no query $Q(x, t^*, c^*)$ will be made.

This leaves queries to `Ver`. Such a query on (a^*, c^*, r) will report if $r = r^*$. Since r^* is uniform in a super-polynomial space, with overwhelming probability all such queries will simply reject. So we could replace all such queries with reject and only negligibly affect the adversary's success probability. But at this point, we have that no queries are made to $Q(x, t^*, c^*)$, and so the probability the adversary wins is negligible. □

On the other hand, we show that if the proof system is an actual proof (that is, it has statistical soundness), then Fiat-Shamir *is* secure:

Theorem 6.2. *FS* is *secure in the AROM for statistically sound proofs, assuming* $|s| \geq |a| + |r| + \omega(\log \lambda)$.

The security of FS for proofs has been explicit conjectured by [BLV03], but a Type 5 impossibility was shown by [BDG+13], showing that FS cannot be proved in the standard model relative to standard assumptions. Thus, an idealized model seems inherent in any justification for security. Before proving Theorem 6.2, we give an idea as to why statistical soundness helps. Notice in our proof of Theorem 6.1, the proof system is not sound against query unbounded attackers, as such an attacker can always generate the first message a regardless of the truthfulness of the statement, and then upon receiving c it can brute-force the message r using `Ver` queries. It is not challenging to come up with oracles that instead provide a statistically sound proof system. Regardless of how it is constructed, such a system would have the property that, for any false x and any first message a, for all but a negligible fraction of possible c, there simply does not exist an valid r. Now, in order for Fiat-Shamir to still be broken, there must be some "bad" c for which there exists a valid r, namely the c resulting from hashing a along with s. Recall that the Fiat-Shamir adversary is efficient and M must be complexity preserving, so the total number of queries to the random oracle O is always polynomial. Since c is the output of O, there is little the adversary/M can do to bias c with polynomial queries. In particular, the set of bad c must be a non-negligible fraction of all c, or else the Fiat-Shamir adversary has a negligible chance of producing such a c. But this then contradicts the statistical security of the original interactive proof system. Theorem 6.2 formalizes this argument. Note that the requirement $|s| \geq |a| + |r| + \omega(\log \lambda)$ appears to be an artifact of our proof technique, and we do not know if it is inherent. We now prove Theorem 6.2:

Proof. Consider a distribution $\mathcal{M}$ and a proof system $(\mathsf{Prov}^{M^O}, \mathsf{Ver}^{M^O})$ that is statistically sound in the $\mathcal{M}$-AROM. Let x be a false statement, and $\mathcal{A}^{O,M^O}$ be an adversary which on inputs s nevertheless outputs (a, c) accepted by $\mathsf{Ver}_{\mathsf{FS}}$ with probability ϵ; in other words, Ver^{M^O} accepts $(x, (a, c = O(s, a), r))$. We will assume for simplicity that M contains a query-forwarding functionality, so that $\mathcal{A}$ does not need direct queries to O. We will also assume without loss of generality that some query $\mathcal{A}$ makes to M will trigger a query on $O(s, a)$, since otherwise the view of $\mathcal{A}$ is independent of c and $\mathcal{A}$ has a negligible success probability.

We construct an *inefficient* adversary $\mathcal{B}$ for $(\mathsf{Prov}^{M^O}, \mathsf{Ver}^{M^O})$. Let ℓ be an upper bound on the length of all queries that $\mathcal{A}$ makes to M, or that M makes to O as a result of an $\mathcal{A}$ query. Let q be the total number of queries that are ultimately made to O. Then $\mathcal{B}$ will query M, O on *all* inputs of length at most ℓ.

Restricting our attention to inputs of length at most ℓ, $\mathcal{B}$ therefore knows O. $\mathcal{B}$ also knows the truth table of M^O, though it does not necessarily know which queries M makes to O. However, it will sample a random $M_0 \leftarrow \mathcal{M}$ conditioned on $M_0^O = M^O$ on all inputs.

$\mathcal{B}$ will now run $\mathcal{A}$ on a random s. It chooses a random $i \in [q]$. For every query z that $\mathcal{A}$ makes to M, $\mathcal{B}$ does the following:

- $\mathcal{B}$ runs $y \leftarrow M_0^O(z)$. If the ith distinct query to O (across all queries $\mathcal{A}$ makes to M) does not occur during this evaluation, then $\mathcal{B}$ gives y to $\mathcal{A}$.
- If the ith distinct query to O happens, call the query (s', a'). If $s = s'$, then $\mathcal{B}$ then sends a' to its challenger, receiving c. It then replaces the value of $O(s, a')$ in its truth table for O with c, resulting in a new oracle O'. $\mathcal{B}$ then evaluates $M_0^{O'}$ using this new truth table. If $s' \neq s$, then $\mathcal{B}$ immediately aborts.

- For all subsequent queries, $\mathcal{B}$ keeps answering with runs $y \leftarrow M_0^{O'}(z)$, using the new truth table.
- When $\mathcal{A}$ outputs (a, r), check that $a = a'$. If not, immediately abort. Otherwise, $\mathcal{B}$ sends r to its challenger.

First note that, by our assumption that $\mathcal{A}$ eventually triggers a query $O(s, a)$, $\mathcal{B}$ aborts with probability $1 - 1/q$, independent of the view of $\mathcal{A}$.

Suppose $\mathcal{B}$ does *not* abort. Notice that the oracle $M_0^{O'}$ seen by $\mathcal{A}$ is identically distributed to the true oracle M^O since it was just re-sampled. Therefore, conditioned on not aborting, $\mathcal{A}$ outputs a valid (a, r) relative to $M_0^{O'}$ with probability ϵ. Factoring in the aborts, we have ϵ/q.

We therefore just need to prove that the r produced by $\mathcal{A}$ is a valid response to $\mathcal{B}$'s challenger, meaning $\mathsf{Ver}^{M^O}(x, (a, c, r))$ accepts. Assume the r is valid for the challenger as seen by $\mathcal{A}$, meaning $\mathsf{Ver}^{M_0^{O'}}(x, (a, c, r))$ accepts. The latter is true with probability ϵ/q. The only way for the different runs of Ver to produce different outputs is if $\mathsf{Ver}^{M^O}(x, (a, c, r))$ makes a query to M that triggers a query on $O(s, a)$. But we can view (a, r) as side-information about s, and the requirement that $|s| \geq |a| + |r| + \omega(\log \lambda)$ means that s has min-entropy given (a, r). As such, over the polynomially-many queries that Ver triggers to O, the probability $O(s, a)$ is amongst them is negligible. Hence, with probability at least $\epsilon/q - \mathsf{negl}$, $\mathcal{B}$'s challenger accepts, violating the statistical security of $(\mathsf{Prov}, \mathsf{Ver})$. □

Remark 6.3. The FS transform is not zero knowledge in the AROM, as the usual zero knowledge proof requires the simulator to be able to program the random oracle, which we disallow due to concerns about Type 4 impossibilities. One option is to use Lindell's transformation [Lin15], which includes a dual-mode commitment that provides a trapdoor for simulation. Another simpler option is to change the way c is computed to $c_i = O(i, s_i, a)$, where c_i is the i-th bit of c, and the CRS is $s = (s_i)_i$ where each s_i is $|a| + |r| + \omega(\log \lambda)$ bits. Now zero knowledge follows from the honest-verifier zero knowledge of $(\mathsf{Prov}, \mathsf{Ver})$: first simulate (a, c, r) for $(\mathsf{Prov}, \mathsf{Ver})$, and then choose random s_i such that $c_i = O(i, s_i, a)$.

Remark 6.4. Our proof above is not amenable to the case of signatures, as we would need a way to answer signing queries without knowing the witness. This is usually accomplished via random oracle programming. Our techniques only allow for programming using an inefficient reduction. But it seems when programming the oracle to answer signing queries, we need the reduction to remain efficient, since an inefficient reduction could have brute-forced the signatures by itself. We therefore leave the signature case as an interesting open question.

7 On Best Possible Hashing

In this section we identify two security properties that are trivially satisfied by random oracles, and each have standard-model instantiations with *different* hash functions. Yet no *single* hash function can satisfy both properties.

The two properties are anti-lossiness (Definition 2.4) and one-wayness against auxiliary input (Definition 2.5).

Anti-lossiness. Recall that anti-lossiness asks that there is no hashing key that makes the function so lossy so as to have predictable outputs when the input is random. Anti-lossiness is a natural property of hash functions, and is likely satisfied by efficient hash functions such as SHA2, where we turn SHA2 into a keyed hash function by simply concatenating the key with the input. After all, the existence of keys/prefixes that allow the output of SHA2 to be predicted would be considered a major weakness of the hash function. It is also easy to construct anti-lossy hash functions information-theoretically, and using public key tools we can even construct collision resistant anti-lossy functions. For example, for key $k = (g, h)$, the map $(x, y) \to g^x h^y$ is collision resistant (assuming discrete log) and anti-lossy. If we treat a random oracle as a keyed function by concatenating the key to the input, random oracles are also trivially anti-lossy.

One-wayness against auxiliary input. Recall that one-wayness against auxiliary input requires that the function remains one-way even against all computationally unpredictable 1-sources. Zhandry [Zha16] constructs such functions under the assumed *exponential* hardness of DDH. A random oracle (treating it as keyed by concatenating the key to the input) also trivially satisfies this notion in the plain ROM: this is just a simple consequence of random oracles being universal computational extractors [BHK13].

7.1 Incompatibility of the definitions

Theorem 7.1. *Assuming VGB obfuscation, there is no hash function H that is both anti-lossy and auxiliary input one-way.*

Proof. The proof is closely related to the insecurity auxiliary input DDH, as shown by [BST16]. Our insight is to identify anti-lossiness as the specific property off DDH that facilitates the proof [BST16]. Let H be any hash function. Our distribution $D(1^\lambda)$ samples a uniformly random x. It then lets $P_x(k, y)$ be the program with x hard-coded, which outputs x if and only if $H(k, x) = y$; otherwise it outputs 0. D then outputs $(x,\ \mathsf{aux} = \mathsf{Obf}(1^\lambda, P_x)\ )$.

Lemma 7.2. *If H is anti-lossy, then D is computationally unpredictable.*

Proof. Suppose towards contradiction that there is an adversary $\mathcal{A}$ for the unpredictability of D. In other words, $\mathcal{A}$ learns x from $\mathsf{Obf}(1^\lambda, P_x)$, for a uniform choice of x. By VGB security, there must therefore be an *inefficient* but query-bounded simulator S such that $S^{P_x}(1^\lambda)$ outputs x with non-negligible probability. Consider any query (k, y) that S makes to P_x. By statistical unpredictability, with overwhelming probability $H(k, x) \neq y$, and so the query response is 0. By a simple hybrid over all queries, with overwhelming probability the answers to all queries are 0, in which case the view of S is independent of x. Hence S cannot output x except with negligible probability, a contradiction. □

We now finish the proof of Theorem 7.1. If given $k, y = H(k, x)$ and aux, we can simply run the program aux on k, y; the result is $P_x(k, y)$, which outputs x since $y = H(k, x)$. Hence, H cannot be one-way against the source D. □

Note that, if H is *not* anti-lossy, then D may be computationally predictable. This is exactly what happens with Zhandry's ELFs. Thus, even though H is not one-way against D, there is no contradiction since D is not a valid source.

We now explain that Theorem 7.1 easily translates to the AROM:

Theorem 7.3. *There is no hash function in the AROM that is both anti-lossy and auxiliary input one-way.*

Proof. The proof closely mirrors the proof of Theorem 7.1. Let H^O be some keyed hash function built from O, which is assumed to be anti-lossy. Let M_{Obf} be the obfuscation oracle from Section 4.3. M_{Obf} is not only iO, but trivially VGB and even VBB. We then define our 1-source $D^O(1^\lambda)$, which samples a uniformly random x. It then lets $P_x^O(k, y)$ be the program with x hard-coded, which outputs x if and only if $H^O(k, x) = y$; otherwise it outputs 0. D^O then outputs $(x,\ \mathsf{aux} = \mathsf{Obf}^O(P_x; r)\ )$ for random coins r. By an essentially identical proof as in the standard model, we see that D^O is computationally unpredictable in the AROM

We now prove that H^O is not auxiliary input one-way with respect to D^O. Indeed, given $k, y = H(k, x)$ and aux, we can run the program aux on k, y; the result is $P_x^O(k, y)$, which outputs x since $y = H(k, x)$. This violates the one-wayness of H under the source D. □

References

[AB12] Prabhanjan Ananth and Raghav Bhaskar. Non observability in the random oracle model. Cryptology ePrint Archive, Report 2012/710, 2012. https://eprint.iacr.org/2012/710.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, August 2007.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2014.

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 182–201. Springer, Heidelberg, March 2013.

[BFM15] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 428–455. Springer, Heidelberg, March 2015.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, August 2013.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *44th FOCS*, pages 384–393. IEEE Computer Society Press, October 2003.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

[BS11] Zvika Brakerski and Gil Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 543–560. Springer, Heidelberg, August 2011.

[BST16] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and UCE. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 542–564. Springer, Heidelberg, January 2016.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469. Springer, Heidelberg, August 1997.

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 389–415. Springer, Heidelberg, January 2016.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

[DOP05] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466. Springer, Heidelberg, August 2005.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

[Gol06] Oded Goldreich. On post-modern cryptography. Cryptology ePrint Archive, Report 2006/461, 2006. https://eprint.iacr.org/2006/461.

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Heidelberg, March 2011.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HK07] Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 466–475. ACM Press, October 2007.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014.

[KM07] Hidenori Kuwakado and Masakatu Morii. Indifferentiability of single-block-length and rate-1 compression functions. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E90-A(10):2301âĂŞ2308, oct 2007.

[KM15] Neal Koblitz and Alfred Menezes. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Report 2015/140, 2015. https://eprint.iacr.org/2015/140.

[KY18] Ilan Komargodski and Eylon Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 259–279. Springer, Heidelberg, April / May 2018.

[Lin15] Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 93–109. Springer, Heidelberg, March 2015.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345. Springer, Heidelberg, May / June 1998.

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 111–126. ACM, January 2013.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017.

[Zha16] Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Heidelberg, August 2016.

[ZZ20] Mark Zhandry and Cong Zhang. Indifferentiability for public key cryptosystems. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 63–93. Springer, Heidelberg, August 2020.