Towards Non-Black-Box Separations of Public Key Encryption and One Way Function

Dana Dachman-Soled *

University of Maryland, danadach@ece.umd.edu

Abstract. Separating public key encryption from one way functions is one of the fundamental goals of complexity-based cryptography. Beginning with the seminal work of Impagliazzo and Rudich (STOC, 1989), a sequence of works have ruled out certain classes of reductions from public key encryption (PKE)—or even key agreement—to one way function. Unfortunately, known results—so called *black-box separations*—do not apply to settings where the construction and/or reduction are allowed to directly access the code, or circuit, of the one way function. In this work, we present a meaningful, *non-black-box* separation between public key encryption (PKE) and one way function.

Specifically, we introduce the notion of BBN^- reductions (similar to the BBNp reductions of Baecher et al. (ASIACRYPT, 2013)), in which the construction E accesses the underlying primitive in a black-box way, but wherein the universal reduction $\mathbb{R}$ receives the efficient code/circuit of the underlying primitive as input and is allowed oracle access to the adversary Adv. We additionally require that the functions describing the number of oracle queries made to Adv, and the success probability of $\mathbb{R}$ are independent of the runtime/circuit size of the underlying primitive. We prove that there is no *non-adaptive,* BBN^- *reduction* from PKE to one way function, under the assumption that certain types of strong one way functions exist. Specifically, we assume that there exists a regular one way function f such that there is no Arthur-Merlin protocol proving that $z \notin \mathsf{Range}(f)$, where soundness holds with high probability over "no instances," $y \sim f(U_n)$, and Arthur may receive polynomial-sized, non-uniform advice. This assumption is related to the average-case analogue of the widely believed assumption $\mathsf{coNP} \not\subseteq \mathbf{NP}/\mathrm{poly}$.

* This work is supported in part by an NSF CAREER Award #CNS-1453045 and by a Ralph E. Powe Junior Faculty Enhancement Award. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

1 Introduction

Complexity-based cryptography seeks to formalize generic assumptions, such as the existence of one way functions or trapdoor functions, and then determine which cryptographic primitives can be constructed from these assumptions. For example, it has been shown that the existence of one way functions implies the existence of pseudorandom generators [**?**], pseudorandom functions [**?**], digitial signatures [**?**,**?**] and symmetric key encryption. For other primitives, such as public key encryption, it is believed that stronger assumptions are necessary. Indeed, a gap between symmetric key and public key encryption schemes also emerges in practice: Practical symmetric key encryption schemes, such as AES, are far more efficient and have proven to be less susceptible to attack, than practical public key encryption schemes, such as RSA. Understanding whether this gap in security and efficiency is inherent seems tied to determining whether public key encryption requires stronger complexity assumptions than one way functions. Unfortunately, even formalizing this question is difficult: We cannot hope to prove that one way function does not imply public key encryption in the *logical* sense, i.e. $\mathsf{OWF} \not\rightarrow \mathsf{PKE}$, since if public key encryption exists then the logical statement $\mathsf{OWF} \rightarrow \mathsf{PKE}$ is always true. Therefore, one approach has been to ask whether there exists a *black-box* reduction of public key encryption to one way function, wherein the construction and security proof (reduction) only access the one way function in an input/output manner, but cannot make use of its code. The answer turns out to be negative as shown by the seminal work of Impagliazzo and Rudich [**?**] (who proved that even key agreement cannot be black-box reduced to one way function) and, in fact, their oracle separation technique was subsequently used to rule out black-box reductions between various primitives such as collision resistant hash functions to one way functions [**?**], oblivious transfer to public key encryption [**?**] and many more. But what about *non-black-box* reductions between these primitives, where the construction/reduction may use the *code* of the underlying primitive?

Pass et al. [**?**] initiated a systematic study of this question, ruling out a type of non-black-box reduction called a *Turing-reductions*—where the code of the underlying primitive is used in an arbitrary manner, but the adversary is used in a black-box manner only—under the assumption that one way functions with very strong properties exist. Briefly, languages coupled with an efficiently samplable distribution over the no instances are considered to be in $\mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$ if there exists an AM (constant-round) protocol that accepts the language, with the relaxation that soundness only needs to hold with high probability over the no instances. For efficiently computable f, Pass et al. [**?**] consider the distributional language $\overline{\mathsf{Range}(f)} = \{z : \forall x \in \{0,1\}^*, f(x) \neq z\}$ along with the distribution $f(U_n)$ over the "No" instances. Their assumption is that there exists an efficiently computable function f such that $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$. Pass et al. [**?**] justify their assumption by arguing that it is a natural average-case analogue of the widely believed assumption $\mathsf{coNP} \not\subseteq \mathsf{AM}$. Based on this assumption, [**?**] rule out various Turing reductions including, reductions from one-way permutations to one-way functions. Additionally, based on other newly introduced complexity

assumptions, Pass et al. [**?**] prove separations among various other primitives. However, none of their results address the case of constructing key agreement (or even public key encryption) from one way function. Separating key agreement from one way function is especially significant, since it implies a separation of public key cryptography from private key cryptography. Indeed, resolving this question is one of the fundamental goals of complexity-based cryptography.

In order to make progress towards this goal, we seek to formalize a meaningful, *non-black-box* separation between one way function and public key encryption (PKE). To the best of our knowledge, the only known separations to date between one way function and public key encryption (PKE) are *oracle* separations. Such separations instantiate the one way function with a random oracle and so do not apply to settings where the construction and/or reduction are allowed to access the code of the one way function. We first define Turing reductions and discuss why it seems hard to rule out all Turing reductions from PKE to one-way function based on the assumption that there exist (classes of) one-way functions f for which $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$. We then introduce and motivate a new, more restricted type of non-black-box reduction, BBN^- reductions, which are related to the BBNp reductions considered in the taxonomy of Baecher et al. [**?**]. Looking ahead, our main theorem will rule out non-adaptive, BBN^- reductions from public key encryption to one-way functions based on the assumption that there exists a regular one-way-function f such that $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\operatorname{poly}}$, where $\mathsf{AM}^{\operatorname{poly}}$ is the non-uniform analogue of AM (i.e. A is allowed to receive polynomial-sized, non-uniform advice).

1.1 Turing Reductions and the Difficulty of Ruling Them Out

We begin by recalling the definition of a type of non-black-box reduction known in the literature as a Turing reduction. The formal definition below will be useful when we define our new class of non-black-box reductions (BBN^- reductions) and compare to the notion of a Turing reduction.

Turing reductions. A Turing reduction from a primitive $\mathcal{Q}$ to a primitive $\mathcal{P}$ is a pair of *oracle* PPT Turing machines $(E, \mathbb{R})$ such that the following two properties hold:

- **Construction.** For every efficient implementation f of primitive $\mathcal{P}$, $E(f)$ implements $\mathcal{Q}$.
- **Reduction.** For every efficient implementation f of $\mathcal{P}$, and every (inefficient) adversary Adv who breaks $E(f)$ with probability $\varepsilon = \varepsilon(n)$, on security parameter n, we have that $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{\varepsilon}, f)$ breaks f with probability $1/t(\max(n, 1/\varepsilon(n)))$ and $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon}, f)$ makes at most $v(\max(n, 1/\varepsilon(n)))$ oracle queries to Adv, for polynomials t, v.

Difficulty of ruling out Turing reductions. To rule out Turing reductions from PKE to one-way function based on the assumption that there exist efficiently computable f for which $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$, one must construct an AM protocol proving $z \notin \mathsf{Range}(f)$ (i.e. that z is "invalid") for any efficiently computable f, assuming there exists a Turing reduction from PKE to one-way

function. The following is the natural way to do this: Let $\mathbb{R}$ be the assumed Turing reduction from PKE to one-way function. The protocol does the following: A emulates the reduction $\mathbb{R}^{\mathsf{Adv}}(f, z)$, using the all-powerful M to respond to queries made to the adversary Adv. Queries made to Adv will be of the form (pk, e) where pk is a public key and e is a ciphertext and in return, M should return the message m encrypted in e, along with a proof (e.g. the coins for Gen and Enc showing that this is a correct decryption). If the emulation of $\mathbb{R}^{\mathsf{Adv}}$ outputs a value x such that $f(x) = z$, then A rejects; otherwise, A accepts. Intuitively, the reason this should work, is that if $\mathbb{R}$ is a "good" reduction, then $\mathbb{R}$ should invert w.h.p. for most $z \sim f(U_n)$, whereas if $z \notin \mathsf{Range}(f)$, no matter what $\mathbb{R}$ does, it cannot invert.

Of course, there is a huge hole in the above argument: The reduction $\mathbb{R}$ may send queries to Adv, that "look like" valid transcripts (pk, e), but actually do not correspond to the output of Gen, Enc on any valid input and randomness. So, we must allow M to claim to A that (pk, e) is invalid, but to prevent M^* from cheating, we must also demand a proof of invalidity. But note that whatever protocol we use to prove that (pk, e) is invalid should not work for proving $z \notin \mathsf{Range}(f)$ for general one way functions f, since this would contradict our assumption. On the other hand, since the AM protocol must work for *every* construction of PKE from one-way function, it is not clear how to restrict the class of functions.

Nevertheless, there is a difference between the two settings: When proving $z \notin \mathsf{Range}(f)$, M^* knows the "statement," i.e., the value of z. On the other hand, *during* the proof of the statement $z \notin \mathsf{Range}(f)$, A samples (pk, e) by running the randomized reduction $\mathbb{R}(f, z)$ and outputting its queries to Adv. Moreover, if M^* cannot distinguish transcripts (pk, e) sampled using the reduction $\mathbb{R}$, from (pk, e) sampled honestly using Gen and Enc, then M^* cannot "cheat." At first glance, it seems that, indeed, the two distributions must be close, since if $\mathbb{R}$'s output is far from the output of Gen and Enc, then Adv can always reject (and thus is useless for inverting f). However, there is a subtle issue here: For Adv to be useful for breaking f, we only need that the output queries of $\mathbb{R}(f, f(U_n))$ (over random variable $f(U_n)$) is close to the output of Gen and Enc; whereas in order to force M^* to behave honestly, we need that the output queries of $\mathbb{R}(f, z)$, *with fixed input* z, are close to the output of Gen and Enc. Thus, in order to force honest behavior from M^*, we would need to show that with high probability over choice of $z \sim f(U_n)$, the queries output by $\mathbb{R}(f, f(U_n))$ are distributed closely to the queries outputted by $\mathbb{R}(f, z)$. In other words, the queries made by $\mathbb{R}(f, z)$ should be (close to) independent of z. But this seems highly implausible since in order for $\mathbb{R}$ to invert z, given oracle access to Adv, a successful $\mathbb{R}$ should embed z in the transcripts (pk, e) it submits to Adv, and so the queries to Adv will clearly depend on z!

Unfortunately, we do not know how to get around this problem for the case of general Turing reductions. However, for the restricted class of non-adaptive, BBN^- reductions, which we introduce next, we will show how to overcome this apparent contradiction.

BBN^- ***reductions.*** A BBN^- reduction from a primitive $\mathcal{Q}$ to a primitive $\mathcal{P}$ is a pair of *oracle* PPT Turing machines $(E, \mathbb{R})$ such that the following two properties hold[1]:

Construction. For every implementation f of primitive $\mathcal{P}$, E^f implements $\mathcal{Q}$.
Reduction. There exist polynomials $t(\cdot), v(\cdot)$ such that: For every efficient implementation f of $\mathcal{P}$, and every (inefficient) adversary Adv who breaks E^f with probability $\varepsilon = \varepsilon(n)$, on security parameter n, we have that $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^\varepsilon, f)$ breaks f with probability $1/t(\max(n, \varepsilon(n)))$ and $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon}, f)$ makes at most $v(\max(n, \varepsilon(n)))$ oracle queries to Adv.

We remark that an implementation of a primitive is any specific scheme that meets the requirements of that primitive (e.g., an implementation of a publickey encryption scheme provides samplability of key pairs, encryption with the public-key, and decryption with the private key).

In the above definition, the construction E makes only black-box calls to f, but the reduction $\mathbb{R}^{\mathsf{Adv}}(f)$ receives the description of f as input and so is *non-black-box*. Allowing only $\mathbb{R}$ access to the code of f already thwarts known techniques (e.g., oracle separations) for proving impossibility results. We also require that the functions describing the number of oracle queries made to Adv, and the success probability of $\mathbb{R}$ are independent of the run-time/circuit size of f.

1.2 Necessity of the Restrictions

The notion of BBN^- reductions is supposed to capture the setting where the construction is "black-box" in the underlying primitive, but the proof is "non-black-box" in the underlying primitive but "black-box" in the adversary. This is a natural subclass of Turing reductions, in which the construction/reduction may both be "non-black-box" in the underlying primitive, but the reduction is "black-box" in the adversary.

However, a careful reader will notice that we placed additional restrictions when defining BBN^- reductions (this was why we called our notion "BBN minus" in that the polynomials $t(\cdot), v(\cdot)$ are independent of the particular function f and so specifically, the polynomials $t(\cdot), v(\cdot)$ must be *independent* of the run-time (i.e. circuit size) of f. Specifically, consider the following alternative definition, which we call BBN':

An Alternative Definition BBN'***:***

Construction. For every implementation f of primitive $\mathcal{P}$, E^f implements $\mathcal{Q}$.
Reduction. For every efficient implementation f of $\mathcal{P}$, and every (inefficient) adversary Adv who breaks E^f with probability $\varepsilon = \varepsilon(n)$, on security parameter n, we have that $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^\varepsilon, f)$ breaks f with probability $1/t(\max(n, \varepsilon(n)))$ and $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon}, f)$ makes at most $v(\max(n, 1/\varepsilon(n)))$ oracle queries to Adv, for polynomials t, v.

[1] We may also consider families of primitives—e.g. families of one-way functions $\mathcal{F}$ with uniform generation algorithms. Here, the generation algorithm is represented as a Turing Machine and each function $f \in \mathcal{F}$ is represented as a circuit.

In the following, we argue that the more restrictive notion of BBN^- is necessary in the following sense: If there exists a Turing reduction from PKE to OWF, then there also exists a BBN' reduction from PKE to OWF. Therefore, ruling out BBN' reductions from PKE to OWF also implies ruling out Turing reductions from PKE to OWF. Since our goal is to relax the notion of Turing reduction in a meaningful way, in order to make progress on this fundamental question, it is necessary to restrict $t(\cdot), v(\cdot)$ as in the definition of BBN^-.

Theorem 1 (Informal) *If there exists a Turing reduction from PKE to (uniform) OWF, then there also exists a* BBN' *reduction from PKE to OWF.*

We sketch the proof of the above theorem.

Proof of Theorem ?? (Sketch): Assume there exists a Turing reduction $(E, \mathbb{R})$ from PKE to one way function, then (using the reduction from one way function to weak one-way function), there also exists a Turing reduction $(E, \mathbb{R})$ from PKE to weak-one-way-function, where an efficient adversary can invert the one way function with probability at most $1 - 1/\operatorname{poly}(n)$, where n is security parameter (i.e. input/output length). We will use this to build a BBN' reduction $(E', \mathbb{R}')$ from PKE to one way function.[2] We first define E': We completely ignore oracle f and set $E' := E(f_{univ})$, where f_{univ} is the "weak" universal one-way function described in [**?**]. Namely, on input Turing machine f' and string x, $f_{univ}(f', x)$ outputs $f'||f'^{|x|^2}(x)$, where $f'^{|x|^2}(x)$ denotes the output of f' after running on input x for $|x|^2$ number of steps. Now, we define the reduction $\mathbb{R}'$: On input (f, y), where f has (polynomial) running time n^c on inputs of length n, and oracle access to adversary Adv breaking E'^f, $\mathbb{R}'^{\mathsf{Adv}}(f, y)$ does the following: Define the new one-way-function f' that runs in time $\widetilde{n}^2$ on inputs of length $\widetilde{n}$ in the following way: f' parses its input as $x||a$, where x has length $\widetilde{n}^{1/c}$ and outputs $f(x) = y$ in time $\widetilde{n}^2$. $\mathbb{R}'$ then runs $\mathbb{R}^{\mathsf{Adv}}$ on inputs $(f_{univ}, (f', y||a))$, where a is a dummy string of length $n^c - n$. Note that the input/output length $\mathbb{R}^{\mathsf{Adv}}$ gets run on is now n^c. Since $\mathbb{R}$ is a good Turing reduction, $\mathbb{R}$ inverts f_{univ} with $1 - 1/\operatorname{poly}(n)$ probability, which means that $\mathbb{R}$ will return an element in $f_{univ}^{-1}((f', y))$ with non-negligible probability. Using this information $\mathbb{R}'$ can then recover an element in $f^{-1}(y)$ with non-negligible probability. However, note that the functions describing the number of times $\mathbb{R}$ runs the adversary Adv and the success probability of $\mathbb{R}$ depend on the input/output length of $(f_{univ}, (f', y||a))$, which is n^c and thus depends on the run time of f. This means that the functions describing the number of times $\mathbb{R}'$ runs Adv and the success probability of $\mathbb{R}'$ depends on the runtime of f. ∎

[2] Note that the argument also holds in the case that the Turing reduction works for a family $\mathcal{F}$ of one way functions f with a uniform generation algorithm. Specifically, if Gen is a uniform, public-coin, generation algorithm that samples a circuit $f \in \mathcal{F}$, then we can construct a single one way function $\widetilde{f}$ that on input randomness r and input x, first runs $\mathsf{Gen}(r)$ to select f, then evaluates $y = f(x)$ and then outputs (r, y).

1.3 Our Main Result

We are now ready to state our main theorem:

Theorem 2 (Informal) *Under the assumption that there exists a regular one-way function f such that the distributional language $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\operatorname{poly}}$, there is no non-adaptive, BBN^{-} reduction from PKE to one way function.*

In the above, $\mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\operatorname{poly}}$ is the same as the class $\mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$, except that A is allowed to receive polynomial-sized, non-uniform advice. Note that our result is restricted to non-adaptive reductions $\mathbb{R}$ which make $v(\max(n, 1/\varepsilon(n)))$ *parallel* oracle queries to the adversary Adv.

We conjecture that using techniques of Akavia et al. [**?**], Theorem **??** can be proven under the assumption that there exists a regular one-way function f such that the distributional language $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}$ (i.e. without requiring the non-uniform advice). The requirement for regularity of f in the assumption comes from our use of the *randomized iterate* (see [**?**]) whose hardness amplification properties only hold for (nearly) regular functions f. Recently, the analysis of the randomized iterate was extended to a more general class of functions called "weakly-regular" functions [**?**]. We conjecture that our results hold for this broader class of functions as well. Extending our results to general one-way functions seems tied to the development of security-preserving hardness amplification techniques for general one-way functions. We leave these as opens problem for future work.

1.4 Our Techniques

A key insight of our work is the relationship between our newly introduced notion of BBN^{-} reductions and the problem of *instance compression*. Instance compression [**?**,**?**,**?**,**?**] is the fundamental complexity-theoretic problem of taking an instance of a hard problem and compressing it into a smaller, equivalent instance, of the same or different problem.[3] The relationship between BBN^{-} reductions and instance compression is the following: The reduction $\mathbb{R}$ takes as input an instance (y, c), where y is a random image of c, and submits queries to Adv, which take the form of transcripts (pk, e) where pk is the public key and e is a ciphertext. Since the public key encryption scheme uses the underlying one-way function in a black-box manner, the size of the transcript (pk, e) must be a fixed polynomial in the security parameter n (i.e. the input-output size of the one-way function). Thus, as long as $\mathbb{R}$ (on input security parameter n) does not query Adv with security parameter $\widetilde{n}$ that is too large and depends on the circuit size (i.e. runtime) of c, then it must be the case that the total length of the messages sent from $\mathbb{R}$ to Adv is *independent* of the size of the circuit c. In order to force $\mathbb{R}$ to

[3] [**?**] showed that strong instance compression algorithms imply a *non-black-box construction* of public key encryption from one-way function. It was later shown that, under standard complexity assumptions, instance compression for certain NP-hard problems is impossible [**?**,**?**], indicating that the approach of [**?**] is unlikely to succeed.

have this behavior, we instantiate Adv in such a way that queries submitted by $\mathbb{R}$ with security parameter which is too large are "useless" due to the restrictions of the BBN^- reduction. Now, in the AM protocol proving statement $z \notin \mathsf{Range}(f)$, instead of using (z, f) itself as the input to $\mathbb{R}$, we construct a new one-way function instance (c, y) (with the same input-output length) using $k = k(n)$ instances $(x_1, y_1), \ldots, (x_k, y_k)$. I.e., $(c, y) \leftarrow \Phi(x_1, y_1), \ldots, (x_k, y_k)$, where Φ is some randomized function, each (x_i, y_i) is an input-output pair of f, and one of the y_i's is set to the common input z. The requirement on $(c, y) \leftarrow \Phi(x_1, y_1), \ldots, (x_k, y_k)$ is that inverting c (i.e. finding x such that $c(x) = y$) implies inverting y_i (i.e. finding x_i such that $f(x_i) = y_i$) with probability $1/\operatorname{poly}(k)$. By choosing $k = k(n)$ to be a sufficiently large polynomial, it is possible to ensure that there is not enough room for all individual instances $y_1, \ldots, y_k$ to be embedded in the interaction with Adv. Thus, the reduction $\mathbb{R}$ itself which takes as input (c, y) and produces queries to Adv can be viewed as an instance compression algorithm. Using techniques of Drucker [**?**] (similar to techniques that appeared previously in [**?**,**?**]), we will now be able to circumvent the problem with the naive attempt to rule out Turing reductions discussed above, which was that with high probability over $z \sim f(U_n)$, the distribution over $\mathbb{R}(f, f(U_n))$ will be far from the distribution over $\mathbb{R}(f, z)$. We elaborate further below on the necessary steps of our proof and in the discussion below, we point out where each restriction we place on the class of reductions is being used:

Eliminating security parameter blow-up. We construct an adversary Adv that has the following property: When the one-way function has input/output length $\widetilde{n}$, Adv flips a coin and returns $\perp$ with probability $1 - 1/\widetilde{n}$. Note that this means that we can replace any reduction $\mathbb{R}$ that on security parameter n makes queries to Adv with extremely large security parameter (i.e. input/output length) greater than $\widetilde{n} := 2 \cdot t(\max(n, 1/\varepsilon(n))) \cdot v(\max(n, 1/\varepsilon(n)))$, with another reduction $\mathbb{R}'$ that simulates all answers of Adv to queries with security parameter greater than $\widetilde{n}$ with $\perp$ without actually making the query. The probability that the view of $\mathbb{R}$ and $\mathbb{R}'$ differs is at most $v(\max(n, 1/\varepsilon(n))) \cdot 1/(2 \cdot t(\max(n, 1/\varepsilon(n))) \cdot v(\max(n, 1/\varepsilon(n))))$ and thus $\mathbb{R}'$ should still succeed with probability at least $1/2t(\varepsilon(n))$. This means that the length of the total output of $\mathbb{R}'$ to Adv depends only on n, but not on the size (runtime) of c and so $\mathbb{R}'$ is indeed a compression function, when we choose appropriate circuit c. Here we use the restriction that $t(), v(), \varepsilon()$ are all independent of the runtime of c.

Designing a circuit-oblivious adversary. The adversary $\mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2, \mathsf{Adv}_3)$ will have the property that $\mathsf{Adv}_1, \mathsf{Adv}_3$ are *efficient* algorithms, whereas Adv_2 is inefficient but *does not require access* to the one-way function c. Looking ahead, M will be used to implement Adv_2 *only*. The fact that Adv_2 does not require access to c is crucial, since otherwise, the size of the interaction would be at least $|c|$ and there would be no compression. The techniques of [**?**,**?**,**?**] are crucial for constructing such Adv. Allowing the construction only black-box access to the underlying one-way function is necessary for this step in the proof, since Adv_2 will essentially emulate the adversary from the black-box separation of OWF and PKE of [**?**,**?**,**?**]. See Section **??**.

Applying instance compression techniques. For a fixed f, denote by $\Phi((x^1, y^1), \ldots, (x^k, y^k))$ the randomized mapping that derives (c, y) from $(x^1, y^1), \ldots, (x^k, y^k)$, where $y^i = f(x^i)$ and view $\mathbb{R} \circ \Phi$ as a compression algorithm. For $z \sim f(U_n)$, we would like to embed $(x^i, y^i) = (x, z)$, where $f(x) = z$, for a random position $i \in [k]$. Call this randomized mapping Φ_z. Using techniques of Drucker [**?**], we will choose Φ so that with high probability over $z \sim f(U_n)$, the distribution over the output of $\mathbb{R} \circ \Phi_z$, denoted $\mathcal{T}(z)$, where a fixed z is embedded in a random position and the remaining inputs are random, is statistically close to the distribution over the output of $\mathbb{R} \circ \Phi$, denoted $\mathcal{T}$ when all (x^i, y^i) are sampled at random. Here also it is crucial to allow the construction only black-box access to the underlying one-way function since otherwise the length of the transcript could depend on the size of c, instead of just the input-output length. We also use here the fact that $\mathbb{R}$'s success probability is independent of the size/run-time of c. This is because the closeness in distributions that we are able to show using techniques of [**?**], will be significantly larger than $1/|c|$. If $\mathbb{R}$ only achieved success probability smaller than $1/|c|$ to begin with, then switching the distributions as discussed above would lead to a "useless" $\mathbb{R}$, which might never succeed in inverting the one-way function.

Designing an AM ***verifier—first stage.*** Unfortunately, even in the "no case," when $z \sim f(U_n)$, A will not be able to sample directly from $\mathcal{T}(z)$ since it will not know a preimage x such that $z = f(x)$. Instead, A will sample from a simulated distribution, denoted by $\widetilde{\mathcal{T}}(z)$. We use techniques of Haitner et al. [**?**] to show that $\widetilde{\mathcal{T}}(z)$ and $\mathcal{T}(z)$ are somewhat close.

Designing an AM ***prover.*** On input an instance z, where z is not in the image of f, we must provide an AM prover who uses $\mathbb{R}$ to prove that z is not in the image. This will yield a contradiction to the existence of $\mathbb{R}$. To construct the AM proof, we use the fact that $\widetilde{\mathcal{T}}(z)$ and $\mathcal{T}$ are somewhat close to allow A to run a rejection sampling protocol with the help of M. This allows A to essentially output transcripts to M that are sampled as in the "honest" distribution $\mathcal{T}$. Using techniques of Bogdanov and Trevisan [**?**] and Akavia et al. [**?**], we can then provide A with non-uniform advice in the form of statistics on the distribution $\mathcal{T}$, which allows him to force M^* to respond to queries honestly

Designing an AM ***verifier—second stage.*** The above steps guarantee that on input (c, y), the reduction $\mathbb{R}^{\mathsf{Adv}}$ (with M assisting A in the simulation of Adv) succeeds in recovering x such that $c(x) = y$ with noticeable probability. However, we must now show that given x, A can also recover x^* such that $f(x^*) = z$ with noticeable probability. Since the circuit c output by Φ is a slight modification of the k-th randomized iterate, defined by Haitner et al. [**?**], we can now leverage hardness amplification properties of the k-th randomized iterate to show that A recovers x^* with $1/\operatorname{poly}$ probability for most $z \sim f(U_n)$ We must also be careful since the argument above guarantees that x can be recovered when the adversary is stateless. It is possible that a stateful M^* can respond in such a way that A recovers x such that $c(x) = y$, but cannot recover x^* such that $f(x^*) = z$. The key to ruling out such a case is that, because of the nature of

public key encryption wherein ciphertexts encrypt either a 0 or a 1, for almost all transcripts output to M^*, there is actually a single "correct" response and we force M^* to respond with this "correct" response with very high probability over the transcripts outputted by the reduction.

1.5 Related Work

In their seminal work, Impagliazzo and Rudich [**?**] ruled out black-box reductions from key agreement to one-way function. Their oracle separation technique was subsequently used to rule out black-box reductions between various primitives such as collision resistant hash functions to one way functions [**?**], oblivious transfer to public key encryption [**?**] and many more. The oracle separation technique cannot be used to rule out non-black-box reductions, since the underlying primitive is modeled as an oracle with an exponentially large description size.

The meta-reduction technique (cf. [**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**]) has been useful for ruling out Turing reductions—reductions where the construction is arbitrary, but the reduction must use the adversary in a black-box manner. Often these techniques are used to give evidence that a construction of primitive P along with a security proof of the above form is impossible under "standard assumptions" (e.g. falsifiable assumptions or non-interactive assumptions). This differs from our setting of separating one-way function from public key encryption, since in this case we can construct public key encryption from most well-studied, concrete assumptions for which we can construct one-way functions (such as factoring, Diffie-Hellman assumptions, and lattice assumptions).

The power of non-black-box usage of the adversary in security reductions has been well-studied since the seminal work of Barak [**?**]. In this case it is well-known that non-black-box technqiues are more powerful than black-box techniques. However, in our work, we are interested in non-black-box use of the underlying *primitive*, as opposed to non-black-box use of the *adversary*. Several recent works have dealt with the systematic study of the power of non-black-box reductions in such settings. These include the aforementioned work of Pass et al. [**?**] as well as a work of Brakerski et al. [**?**], which, among other results, addresses the question of whether zero knowledge proofs can help to construct key agreement from one-way function. However, the results of Brakerski et al. hold only in an oracle setting, where an oracle is added to simulate the power of a zero-knowledge proof. Baecher et al. [**?**] gave a taxonomy of black-box and non-black-box reductions. Indeed, the term BBN that we use is borrowed from Baecher et al. [**?**], who used BBN to indicate reductions wherein the construction uses the primitive in a **B**lack-box manner, the reduction uses the adversary in a **B**lack-box manner, but the reduction uses the primitive in a **N**on-black-box manner. Our notion of BBN^- differs from the notion of Baecher et al. [**?**] in that we require the reduction $\mathbb{R}$ to be universal, but allow $\mathbb{R}$ to receive the description of the code/circuit of f as input. Moreover, we allow the query complexity and success probability of $\mathbb{R}$ to depend on the success probability of the adversary Adv, but require it to be independent of the run-time/circuit size of f.

2 Preliminaries and Background

Notation. We use capital letters for random variables, standard letters for variables and calligraphic letters for sets. We adopt the convention that when the same random variable appears multiple times in an expression, all occurrences refer to the same instantiation. Given a distribution X and an event E, we denote by $X \mid E$ the conditional distribution over X, conditioned on the event E occurring. Let X be a random variable taking values in a finite set $\mathcal{U}$. If $\mathcal{S}$ is a subset of $\mathcal{U}$, then $x \sim \mathcal{S}$ means that x is selected according to the uniform distribution on $\mathcal{S}$. We write U_n to denote the random variable distributed uniformly over $\{0,1\}^n$ and $U_{[0,1]}$ to denote the continuous random variable distributed uniformly over $[0,1]$. In general, for a finite set S, we denote by U_S the uniform distribution over S.

Two distributions X and Y over $\mathcal{U}$ are ε close, denoted $\Delta(X,Y) \leq \varepsilon$, if $\frac{1}{2}\sum_{x\in\mathcal{U}} |\Pr_X[x] - \Pr_Y[x]| \leq \varepsilon$. For a set $\mathcal{S} \subseteq \mathcal{U}$, we denote by $\Pr_X[S] := \sum_{x\in\mathcal{S}} \Pr_X[x]$, i.e. the weight placed on $\mathcal{S}$ by the distribution X.

For functions $f : \{0,1\}^n \to \{0,1\}^n$ and $y \in \{0,1\}^n$, we denote by $f(U_n)$ the distribution induced by f operating on U_n and we denote by $f^{-1}(y)$ the set $f^{-1}(y) := \{x \in \{0,1\}^n : f(x) = y\}$. For a distribution X with (implicit) sampling algorithm Samp, that takes n coins, we denote by $X(r)$ for $r \in \{0,1\}^n$, the output x of $\mathsf{Samp}(r)$. For an element x in the support of X, we denote by $X^{-1}(x)$ the set of random coins $r \in \{0,1\}^n$ such that $X(r) = x$.

Let $\mathcal{C} = \{\mathcal{C}_{k,n}\}$ be a parametrized collection of uniformly generated polynomially-sized circuits, indexed by $n \in \mathbb{N}$ and $k = k(n) = \mathrm{poly}(n)$. For a fixed (n,k) pair, let $C_{k,n}$ denote the random variable representing the choice of circuit $c_{k,n} \sim \mathcal{C}_{k,n}$, where $\mathcal{C}_{k,n}$ is a family of one-way functions. We require that with probability 1, $C_{k,n}$ implements a one-way function.

Definition 1 (BBN^- reduction from PKE to OWF). *A BBN^- reduction from public key encryption (PKE) to one-way function (OWF) is a pair of* oracle *PPT Turing machines $(E,\mathbb{R})$ with the following properties:*

- ***Construction.*** *With all but negligible probability over $C_{k,n}$, $E^{C_{k,n}}(1^n)$ implements a PKE scheme.*
- ***Reduction.*** *There exist polynomials $t(\cdot), v(\cdot)$ such that: For every (inefficient) adversary Adv who, with probability $\varepsilon_1 = \varepsilon_1(n) = 1/\mathrm{poly}(n)$ over $c_{k,n} \sim C_{k,n}$, breaks $E^{c_{k,n}}(1^n)$ with probability $\varepsilon_2 = \varepsilon_2(n) = 1/\mathrm{poly}(n)$, we have:*

$$\Pr_{c_{k,n}\sim C_{k,n}}\left[\Pr\left[\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{\frac{1}{\varepsilon_2}}, c_{k,n}, c_{k,n}(U_n)) \in c_{k,n}^{-1}(c_{k,n}(U_n))\right] \geq \frac{1}{t(\max(n, \frac{1}{\varepsilon_2(n)}))}\right] \geq \varepsilon_1,$$

  and $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon}, c_{k,n}, y)$ makes at most $v(\max(n, 1/\varepsilon_2(n)))$ oracle queries to the adversary Adv.

Definition 2 (BBN^- reduction from PKE to $(1-\delta/2)$-weak one way function). *A BBN^- reduction from public key encryption (PKE) to $(1-\delta)$-weak one-way function (for $q = \mathrm{poly}(n)$) is a pair of* oracle *PPT Turing machines $(E,\mathbb{R})$ with the following properties:*

Construction. *With all but negligible probability over $C_{k,n}$, $E^{C_{k,n}}(1^n)$ implements a PKE scheme.*

Reduction. *There exists a polynomial $v(\cdot)$ such that: For every (inefficient) adversary Adv who, with probability $\varepsilon_1 = \varepsilon_1(n) = 1/\operatorname{poly}(n)$ over $c_{k,n} \sim C_{k,n}$, breaks $E^{c_{k,n}}(1^n)$ with probability $\varepsilon_2 = \varepsilon_2(n) = 1/\operatorname{poly}(n)$, we have:*

$$\Pr_{c_{k,n}\sim C_{k,n}}\left[\Pr\left[\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon_2}, c_{k,n}, c_{k,n}(U_n)) \in c_{k,n}^{-1}(c_{k,n}(U_n))\right] \geq 1 - \delta/2\right] \geq \varepsilon_1,$$

and $\mathbb{R}^{\mathsf{Adv}}(1^n, 1^{1/\varepsilon_2}, c_{k,n}, y)$ makes at most $v(\max(n, 1/\varepsilon_2(n)))$ oracle queries to the adversary Adv.

Definition 3 (Non-adaptive Reductions $\mathbb{R}$). *The reduction $\mathbb{R} = (\mathbb{R}_1, \mathbb{R}_2)$ is* non-adaptive *if it interacts with the adversary Adv in the following way:*

- *On input $(1^n, 1^{1/\varepsilon_2}, c_{k,n}, y)$ and random coins, $\mathbb{R}_1$ produces a transcript tr consisting of $v(\max(n, 1/\varepsilon_2(n)))$ parallel queries to Adv, as well as the intermediate state st.*
- *On input tr, Adv returns responses $d_1, \ldots, d_{v(\max(n,1/\varepsilon_2(n)))}$. $\mathbb{R}_2(\mathsf{st}, d_1, \ldots, d_{v(\max(n,1/\varepsilon_2(n)))})$ returns either x such that $c_{k,n}(x) = y$ or returns $\perp$.*

For fixed $(\mathsf{tr}, \mathsf{st})$ pair, we also denote the output of $\mathbb{R}_2$ with respect to an oracle Adv and a fixed $(\mathsf{tr}, \mathsf{st})$ output by $\mathbb{R}_1$, by $\mathbb{R}^{\mathsf{Adv}}(c, y, \mathsf{tr}, \mathsf{st}; r)$ or $\mathbb{R}^{\mathsf{Adv}}(c, y, \mathsf{tr}, \mathsf{st})$ (depending on whether the coins of $\mathbb{R}_2$ are explicit or implicit). Note that in the above, r denotes the coins used by $\mathbb{R}_2$ only (and not the coins of $\mathbb{R}_1$ or Adv).

Constant-round interactive protocols with advice. An interactive protocol with advice consists of a pair of interactive machines $\langle P, V\rangle$, where P is a computationally unbounded prover and V is a PPT verifier which receive a common input x and advice string a. Feigenbaum and Fortnow [**?**] define the class $\mathsf{AM}^{\mathrm{poly}}$ as the class of languages L for which there exists a constant c, a polynomial p and an interactive protocol $\langle P, V\rangle$ with advice such that for every n, there exists an advice string a of length $p(n)$ such that for every x of length n, on input x and advice a, $\langle P, V\rangle$ produces an output after c rounds of interaction and, for small constant ε':

- If $x \in L$, then $\Pr[\langle P, V\rangle$ accepts x with advice $a] \geq 1 - \varepsilon'$.
- If $x \notin L$, then for every prover P^*, $\Pr[\langle P^*, V\rangle$ accepts x with advice $a] \leq \varepsilon'$.

It was shown by [**?**] that $\mathsf{AM}^{\mathrm{poly}}$ is equal to $\mathbf{NP}/\mathrm{poly}$. Thus, $\mathsf{coNP} \subseteq \mathsf{AM}^{\mathrm{poly}}$ implies $\mathsf{coNP} \subseteq \mathbf{NP}/\mathrm{poly}$, which gives $\Sigma_3 = \Pi_3$ [**?**]. We use the terms M, "prover" and P (resp. A, "verifier" and V) interchangeably.

Definition 4. *A distributional language (L, D) is in $\mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\mathrm{poly}}$ if for every inverse polynomial q, there exists an AM (i.e., constant-round public-coin) protocol (P, V) where A receives advice of length polynomial in the input length such that, for small constant ε':*

- ***Completeness:*** *If $x \in L$, $\Pr[\langle P, V\rangle(x) = 1] \geq 1 - \varepsilon'$.*

– ***Soundness:*** *For every $n \in \mathbb{N}$ and every machine P^*, with probability $1 - q(n)$, and $x \in \{0,1\}^n$ sampled from D_n conditioned on $x \notin L$ satisfies* $\Pr[\langle P^*, V\rangle(x) = 1] \leq \varepsilon'$.

Our protocols will use the AM protocol $\mathsf{RandSamp}^m$ (the multi-query variant of RandSamp) with the following properties as a subroutine. RandSamp has been used extensively in the literature; the formalization below is due to [**?**].

Lemma 3 *Let $w = g(r)$ for a $\mathrm{poly}(n)$-time computable, randomized function g and random coins r. Assume w has bit length $\widehat{n}$. Then there exists an AM protocol $\mathsf{RandSamp}^m$ with an efficient verifier V that gets as input a security parameter 1^n, $\delta' = 1/\mathrm{poly}(n)$ (as the approximation and soundness parameter), $s_1, \ldots, s_m$ (as size of $f^{-1}(w_1), \ldots, f^{-1}(w_m)$) such that for all $i \in [m]$, $s_i \in (1 \pm \lambda)|f^{-1}(w_i)|$ (for $\lambda = \mathrm{poly}(1/m, 1/(\widehat{n} \cdot m), \delta')$ and returns $(r_1, \ldots, r_m)$ such that:*

– ***Completeness:*** *There is a prover strategy (the honest prover) s.t. V aborts with probability at most δ.*
– ***Soundness:*** *For any prover P^* either*
  • *$\langle P^*, V\rangle$ aborts with probability $1 - \delta'$ OR*
  • *$\Delta((U_{f^{-1}(w_1)}, \ldots, U_{f^{-1}(w_m)}), (r_1, \ldots, r_m))) \leq \delta' + \Pr[\langle P^*, V\rangle$ aborts$]$.*

The following fact will be useful when protocol $\mathsf{RandSamp}^m$ is employed:

Fact 4 *Let X, Y be random variables distributed over the set $\mathcal{S} \cup \{\perp\}$ such that $\Pr[Y = \perp] = 0$ and $\Delta(X, Y) \leq \Pr[X = \perp] + \delta'$. Then for any event $T \subset \mathcal{S}$ it holds that:*

$$\Pr[X \in T] = \Pr_{x \sim X}[x \neq \perp \wedge x \in T] \leq \Pr[Y \in T] + \delta'.$$

and so

$$\Pr_{x \sim X|(x \neq \perp)}[x \in T] \leq (\Pr[Y \in T] + \delta') \cdot \frac{1}{\Pr_{x \sim X}[x \neq \perp]}.$$

Definition 5 (Enhanced Randomized Iterate). *Let $f : \{0,1\}^n \to \{0,1\}^n$, let $\mathcal{H}$ be a family of pairwise-independent length-preserving hash functions over strings of length n and let $\widehat{\mathcal{H}}$ be a family of $p' \cdot p_q(\widetilde{n}) + p(\widetilde{n})$-wise independent length-preserving hash functions (where $p', p_q(\widetilde{n}), p(\widetilde{n})$ are polynomials in n that will be defined later) over strings of length n. Define the k-th* enhanced randomized iterate *$F : \{0,1\}^n \times \mathcal{H}^{k-1} \times \widehat{\mathcal{H}}^2 \to \{0,1\}^n$ as*

$$F(x, \overline{h}, \widehat{h}_1, \widehat{h}_2) = \widehat{h}_2(f(h_{k-1}(f(h_{k-2}(\cdots(f(\widehat{h}_1(x)))\cdots))))).$$

We denote by H_j (resp. $\hat{H}_b$, $b \in \{1,2\}$) random variables uniformly distributed over $\mathcal{H}$ (resp. $\hat{\mathcal{H}}$).

Let $c_{k,n} = c_{k,n}(\cdot, \overline{h} = h_1, \ldots, h_{k-1}, \widehat{h}_1, \widehat{h}_2)$ denote the circuit which has $\overline{h}, \widehat{h}_1, \widehat{h}_2$ hardwired and on input x computes $y = F(x, \overline{h}, \widehat{h}_1, \widehat{h}_2)$. Let $\mathcal{C}_{k,n}$ denote the set of circuits $c_{k,n}$ obtained when taking $h_1, \ldots, h_{k-1} \in \mathcal{H}, \widehat{h}_1, \widehat{h}_2 \in \hat{\mathcal{H}}$. Let $C_{k,n}$ be the random variable defined as $C_{k,n} = c_{k,n}(\cdot, H_1, \ldots, H_{k-1}, \hat{H}_b, \hat{H}_b)$.

Lemma 5 ([?]) *For $i \in [k]$, let $c^i_{k,n} = c^i_{k,n}(\cdot, \overline{h} = h_1, \ldots, h_{i-1}, \widehat{h}_1)$ denote the circuit which has $\overline{h}, \widehat{h}_1$ hardwired and on input x computes $y^i = \mathbb{Y}^i(c_{k,n}, x) = F^i(x, \overline{h}, \widehat{h}_1)$. Let the random variable $C^i_{k,n}$ denote the distribution over circuits $c^i := c^i_{k,n}$ as above.*
Then for any set $\mathcal{L} \subseteq \{0,1\}^n \times \mathcal{C}^i_{k,n}$ with

$$\Pr[(C^i_{k,n}(U_n), C^i_{k,n}) \in \mathcal{L}] \geq \delta,$$

it holds that

$$\Pr[(f(U_n), C^i_{k,n}) \in \mathcal{L}] \geq \frac{\delta^2}{i}.$$

We now describe a transformation (folklore, formalized by Haitner et al. [?]), of an arbitrary one-way function into a length-preserving one-way function.

Lemma 6 *Let $f : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a $(T = T(n), \varepsilon = \varepsilon(n))$-OWF and let $\mathcal{H}$ be an efficient family of 2^{-2n}-almost pairwise-independent hash functions from $\{0,1\}^{\ell(n)}$ to $\{0,1\}^{2n}$. We define $\overline{f}$ as*

$$\overline{f}(x_a, x_b, h) = (h(f(x_a)), h),$$

where $x_a, x_b \in \{0,1\}^n$ and $h \in \mathcal{H}$. Then $\overline{f}$ is a length-preserving $(T - n^{O(1)}, \varepsilon + 2^{-n+1})$-one-way function.

If the original function f is regular, then the output function $\overline{f}$ is *nearly* regular: There is some fixed s such that with all but negligible probability over $y \sim \overline{f}(U_{2n}, H)$, the number of pre-images of y is exactly s. It turns out that nearly regular functions are sufficient for all of our results.

3 The circuit-oblivious adversary Adv

Let $E^f = (\mathsf{Gen}^f, \mathsf{Enc}^f, \mathsf{Dec}^f)$ be a public key encryption scheme making oracle calls to one-way function f. Assume polynomial $p_q(n)$ is an upperbound on the total number of queries made by Gen^f, Enc^f, Dec^f on input security parameter n and message of length n. We consider the following two distributions corresponding to sampling the function f from two different distributions.

In the following, $\mathcal{F}_n$ denotes the set of all functions from $\{0,1\}^n \to \{0,1\}^n$.

Note that when $c \sim \mathcal{C}_{k,n}$ is fixed, we write $\mathcal{E}^c$ to denote the distribution $\mathcal{E}^C$, with a fixed oracle c (whereas C denotes a random variable).

We next describe a modification (folklore and formally proved in [?]) of the well-known Eve algorithm, which is tailored for breaking public key encryption in the random oracle model. The advantage of this $\mathsf{Eve} = (\mathsf{Eve}_1, \mathsf{Eve}_2, \mathsf{Eve}_3)$ algorithm is that $\mathsf{Eve}_1, \mathsf{Eve}_3$ are polynomial-time and Eve_2 is inefficient but *does not* require oracle access to $\mathcal{O}$.

Eve runs on transcripts of the form (pk, e), where pk is the public key and e is the ciphertext. Eve's goal is to correctly decrypt e. Sotakova [?] proves the existence of an Eve with the following properties:

Eve_1 is an *efficient* oracle algorithm which takes input pk and outputs $\mathcal{Q}_{\mathsf{Eve}}$:

Distribution $\mathcal{E}^C$

To sample from this distribution:

- $c \sim \mathcal{C}_{k,n}$; $m \sim \{0,1\}$.
- $(pk, sk) \sim \mathsf{Gen}^{C(\cdot)}(1^n)$; $\mathsf{e} \sim \mathsf{Enc}^c_{pk}(m)$.
- Output $(c, m, sk, pk, \mathsf{e})$.

Distribution $\mathcal{E}^{\mathcal{O}}$

To sample from this distribution:

- $\mathcal{O} \sim \mathcal{F}_n$; $m \sim \{0,1\}$.
- $(pk, sk) \sim \mathsf{Gen}^{\mathcal{O}(\cdot)}(1^n)$; $\mathsf{e} \sim \mathsf{Enc}^{\mathcal{O}}_{pk}(m)$.
- Output $(\mathcal{O}, m, sk, pk, \mathsf{e})$.

- Initialize $\mathcal{Q}_{\mathsf{Eve}} := \emptyset$. Choose $\hat{p}$ random strings $r^1, \ldots, r^{\hat{p}}$ and messages $m^1, \ldots, m^{\hat{p}}$.
- For $1 \leq i \leq \hat{p}$, run $\mathsf{Enc}^{\mathcal{O}}_{pk}(m^i; r^i)$. Add all queries and responses to $\mathcal{Q}_{\mathsf{Eve}}$. Let $p(n) = \text{poly}(n)$ be the total number of queries made. $p(\cdot)$ depends only on $p_q(n)$ and the desired success probability $1 - \delta/8$.

Eve_2 takes $(pk, \mathcal{Q}_{\mathsf{Eve}})$ as input and outputs $[(\mathcal{I}_i, r_i)]_{i \in [p'(n)]}$ (note that Eve_2 *does not* have oracle access):

- Return p' number of elements $\{(\mathcal{I}_1, r_1), \ldots, (\mathcal{I}_{p'}, r_{p'})\}$ chosen uniformly at random from the set $\mathcal{S}(pk, \mathcal{Q}_{\mathsf{Eve}}) := \{(\mathcal{I}, r) \mid \mathsf{Gen}^{\mathcal{I}}(r) = (*, pk) \wedge \mathsf{Eve}_1^{\mathcal{I}}(pk) = \mathcal{Q}_{\mathsf{Eve}} \wedge |\mathcal{I}| = p_q(n) + p(n)\}$. [4]

Eve_3 is an *efficient* oracle algorithm which takes $[(\mathcal{I}_i, r_i, \mathsf{e})]_{i \in [p']}$ as input and outputs a bit d.

- For $i \in [p']$, run $\mathsf{Gen}^{\mathcal{I}_i}(r_i)$ to generate a (sk_i, pk_i)-pair and compute $\widetilde{d}_i := \mathsf{Dec}^{\mathcal{I}_i, \mathcal{O}}_{sk}(\mathsf{e})$. By this notation we mean that whenever Dec queries the oracle, if the query is in $\mathcal{I}$, respond according to $\mathcal{I}$. Otherwise, respond according to $\mathcal{O}$.
- Given the resulting set of decryptions $\{\widetilde{d}_1, \ldots, \widetilde{d}_{p'}\}$, let num_0 denote the number of decryptions equaling 0 and num_1 denote the number of decryptions equaling 1. Let $b = 0$ if $\mathsf{num}_0 > \mathsf{num}_1$ and $b = 1$ otherwise.
- If $V := \mathsf{num}_0/p' \in [3/8 + (\ell - 1)/4p'', 3/8 + (\ell + 1)/4p'']$, return $d := 0$. Otherwise, return $d := b$.

We define parameters p', p'', ℓ in the full version. The exact setting will depend on properties of the given BBN^- reduction $\mathbb{R}$.

We next turn to proving success of the adversary.

Lemma 7 ([?], restated) *For $(\mathcal{O}, m, sk, pk, \mathsf{e}) \sim \mathcal{E}^{\mathcal{O}}$, $\mathsf{Eve}^{\mathcal{O}}(pk, \mathsf{e})$ outputs m with probability at least $1 - \delta/8$.*

The basic intuition is the following: Given the first message pk sent from receiver to the sender, w.h.p, the set $\mathcal{Q}_{\mathsf{Eve}}$ will contain all queries made by the sender when computing the second message (the ciphertext e) with probability greater than some threshold $1/p_{th}(n)$. Now, we sample a view for the receiver

[4] $\mathcal{I}$ is an ordered set of $p_q(n) + p(n)$ length n strings. Whenever Gen or Eve_1 make a query, if the query has not been made before, the next string is used to respond to the query. If the query has previously been made, the same string is returned.

consistent with $(pk, \mathcal{Q}_{\mathsf{Eve}})$, which will contain a secret key sk and use this secret key sk to decrypt the *real* ciphertext e sent by the real sender. Loosely speaking, sk should only decrypt e "incorrectly" if there is a query q to the random oracle that is answered *inconsistently* in the sampled receiver's view and the real sender's view. However, note that any individual query q that is made in the sampled receiver's view but is not contained in $\mathcal{Q}_{\mathsf{Eve}}$, is made by the real sender with probability less than $1/p_{th}(n)$. Now, since we choose p_{th} far larger than the number of queries contained in the receiver's view, it is unlikely that there are *any* queries in the sampled receiver's view that were also made by the sender, but do not appear in $\mathcal{Q}_{\mathsf{Eve}}$. Thus, w.h.p, there are no queries q answered inconsistently in the sampled receiver's view and real sender's view and thus with high probability, the sampled sk decrypts e correctly.
We now describe the actual adversary $\mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2, \mathsf{Adv}_3)$:

- Adv_1**:** On input pk, and oracle access to c, Adv_1 computes $\mathcal{Q}_{\mathsf{Eve}} \leftarrow \mathsf{Eve}_1^c(pk)$, where Eve's queries are answered according to c (instead of the random oracle $\mathcal{O}$). Adv_1 outputs $\mathcal{Q}_{\mathsf{Eve}}$.
- Adv_2**:** On input $(pk, \mathcal{Q}_{\mathsf{Eve}})$, Adv_2 runs $\mathsf{Eve}_2(pk, \mathcal{Q}_{\mathsf{Eve}})$ and outputs $[(\mathcal{I}_i, r_i)]_{i\in[p'(n)]}$.
- Adv_3**:** On input $[(\mathcal{I}_i, r_i, \mathsf{e})]_{i\in[p'(n)]}$, Adv_3 runs $\mathsf{Eve}_3^c([(\mathcal{I}_i, r_i, \mathsf{e})]_{i\in[p'(n)]})$ where Eve's queries are answered according to c (instead of the random oracle $\mathcal{O}$). Adv_3 flips a coin and outputs $\perp$ with probability $1 - 1/n$. With probability $1/n$, Adv outputs the same bit d that is outputted by Eve_3^c.

We purposely "weaken" the adversary, by defining Adv such that it outputs $\perp$ with probability $1 - 1/n$—where n is the input/output length of the one-way function—in order to argue that queries made by the reduction, $\mathbb{R}$, to Adv with security parameter n set too large are "useless." See Section **??** for further discussion. We next turn to proving success of the adversary:

Lemma 8 *For $(c, m, sk, pk, \mathsf{e}) \sim \mathcal{E}^C$, d computed by $\mathsf{Adv}^c(pk, \mathsf{e})$ is equal to m with probability at least $1 - \delta/4$.*

Intuitively, Lemma **??** holds since Adv^c makes at most $p(n)+p'\cdot p_q(n)$ queries and so since $\hat{h}_1, \hat{h}_2$ are $p(n)+p'\cdot p_q(n)$-wise independent, the view of the adversary is nearly the same when interacting with a random oracle $\mathcal{O}$ or with the randomly sampled circuit C. For the full proof, see the full version.

Now, using Markov's inequality and the fact that Adv_3 tosses a coin independently of all its other coins to decide whether to output $\perp$ at the final stage with probability $1 - 1/n$, we have that:

Corollary 9 *With probability $\varepsilon_1 := 1 - \delta/2$ over choice of $c \sim C$, we have that for $(m, sk, pk, \mathsf{e}) \sim \mathcal{E}^c$, the output of $\mathsf{Adv}^c(pk, \mathsf{e})$ is equal to m with probability is at least $\varepsilon_2 := \delta/4n$.*

4 The mapping Φ

Instead of sampling $c \sim C_{k,n}$, $x \sim U_n$, and outputting $(c, x, y := c(x))$, we can alternatively sample $(x^1, y^1), \ldots, (x^k, y^k) \sim (U_n, f(U_n))$ and $r \sim \{0,1\}^*$, set $(x, c, y) := \Phi((x^1, y^1), \ldots, (x^k, y^k); r)$, for Φ defined below:

The randomized mapping $\Phi((x^1, y^1), \ldots, (x^k, y^k))$

- Sample $x \sim U_n$, $\widehat{h}_2 \sim \widehat{\mathcal{H}}$.
- Sample $h_1, \ldots, h_{k-1} \sim \mathcal{H}$, such that $h_i(y^i) = x^{i+1}$, for $i \in [k-1]$; $\widehat{h}_1 \sim \widehat{\mathcal{H}}$ such that $\widehat{h}_1(x) = x^1$.
- Return the tuple (x, c, y) where $c = c_{k,n}(\cdot, h_1, \ldots, h_{k-1}, \widehat{h}_1, \widehat{h}_2)$ and $y = c(x)$.

It is straightforward to see that the two sampling methods described above induce the same distribution. We additionally introduce the notation Φ_2 to denote the second and third coordinates of the output of Φ (i.e. (c, y)).

5 Useful Distributions

For public key encryption scheme $E^{\mathcal{O}} = (\mathsf{Gen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$, relative to random oracle $\mathcal{O}$, the following distribution (Figure **??**) corresponds to sampling a partial random oracle and running Gen.

To sample from distribution $\mathcal{PK}_n$:

- Sample a set $\mathcal{I}$ of $p' \cdot q_p(n) + p(n)$ random strings of length n. Sample randomness $r \sim \{0,1\}^*$.
- Compute $(pk, sk) := \mathsf{Gen}^{\mathcal{I}}(1^n; r)$, where queries are answered using $\mathcal{I}$.
- Compute $\mathcal{Q}_{\mathsf{Eve}} := \mathsf{Adv}_1^{\mathcal{I}}(pk)$, where queries are answered using $\mathcal{I}$.
- Output $(pk, \mathcal{Q}_{\mathsf{Eve}})$.

Fig. 2: The distribution $\mathcal{PK}_n$.

We assume that security parameter n can be determined given the generated pk. We slightly abuse notation and for a fixed $(pk, \mathcal{Q}_{\mathsf{Eve}})$, we denote by $\mathcal{PK}^{-1}(pk, \mathcal{Q}_{\mathsf{Eve}})$ the set of pairs $(\mathcal{I}, r)$ that yield output $(pk, \mathcal{Q}_{\mathsf{Eve}})$ when sampling from $\mathcal{PK}_n$, for appropriate n.

For each of the following distributions χ, we refer by χ_2 to the marginal distribution over the final coordinate, the transcript $\widetilde{\mathsf{tr}}$. For marginal distributions (e.g. the marginal distribution over the second, third and sixth coordinates) we use full-length tuples with $*$ symbols in the "don't care" positions (e.g. $(*, c, y, *, *, i, *, *) \sim \mathcal{T}$ or $\Pr_{\mathcal{T}}[(*, c, y, *, *, i, *, *)]$). To denote the distribution χ, conditioned on one of the tuple coordinates fixed to some value v, we write $\chi \mid v$, where it is understood from context which coordinate is fixed (e.g. $\mathcal{T} \mid c$ means that the second coordinate is fixed to constant c). Note that if χ is a distribution over tuples with t number of coordinates, then $\chi \mid v$ is a distribution over tuples with $t-1$ number of coordinates.

Henceforth, we fix a particular BBN^- reduction $\mathbb{R}$ with parameters $(v(\cdot), t(\cdot))$ and use the particular adversary Adv with success probability $(\varepsilon_1 = 1 - \delta/4, \varepsilon_2 =$

δ) defined in Section **??**. We denote by $v'(\cdot), t'(\cdot)$ the following polynomials: $v'(n) := v(\max(n, 1/\varepsilon_2(n))$ and $t'(n) := t(\max(n, 1/\varepsilon_2(n)))$.

We next define the distributions $\mathcal{T}$ and $\widetilde{\mathcal{T}}$ in Figures **??** and **??**.

To sample from distribution $\mathcal{T}$:
- $i \sim [k]$; $x^1, \ldots, x^k \sim U_n$. Set $y^1 := f(x^1), \ldots, y^k := f(x^k)$ Choose $r, r' \sim \{0,1\}^*$.
- Set $(x, c, y) := \Phi((x^1, y^1), \ldots, (x^k, y^k); r)$, where $c := c(\cdot, h_1, \ldots, h_{k-1}, \widehat{h}_1, \widehat{h}_2)$, $c^i := c^i(\cdot, h_1, \ldots, h_{i-1}, \widehat{h}_1)$.
- Compute $(\mathsf{st}, \mathsf{tr}) = \mathbb{R}_1(c, y; r')$, where $\mathsf{tr} = ((pk_1, \mathsf{e}_1), \ldots, (pk_{v'(n)}, \mathsf{e}_{v'(n)}))$.
- For each $j \in [v'(n)]$, set $\mathcal{Q}^j_{\mathsf{Eve}} = \mathsf{Adv}^c_1(pk_j)$.
- Ouput $(x, c, y, c^i, y^i, i, \mathsf{st}, \widetilde{\mathsf{tr}})$, where $\widetilde{\mathsf{tr}} = ((pk_1, \mathcal{Q}^1_{\mathsf{Eve}}, \mathsf{e}_1), \ldots, (pk_{v'(n)}, \mathcal{Q}^{v'(n)}_{\mathsf{Eve}}, \mathsf{e}_{v'(n)}))$.

Fig. 3: The distribution $\mathcal{T}$.

Let $\mathsf{num}_{\mathcal{T}}$ be the number of random coins to sample from $\mathcal{T}$. Let $\mathsf{N}_{\mathcal{T}} := 2^{\mathsf{num}_{\mathcal{T}}}$.

To sample from distribution $\widetilde{\mathcal{T}}$:
- $i \sim [k]$, $h_1, \ldots, h_{k-1} \sim \mathcal{H}$, $\widehat{h}_1, \widehat{h}_2 \sim \widehat{\mathcal{H}}$, $r' \sim \{0,1\}^*$. Choose $x^* \sim U_n$, $y^i := f(x^*)$.
- Set $y := F_i^{k-i}(y^i, h_{i+1}, \ldots, h_{k-1}, \widehat{h}_2)$. Set the circuit $c := c(\cdot, h_1, \ldots, h_{k-1}, \widehat{h}_1, \widehat{h}_2)$ and $c^i = c(\cdot, h_1, \ldots, h_{i-1}, \widehat{h}_1)$.
- Compute $(\mathsf{st}, \mathsf{tr}) = \mathbb{R}_1(c, y; r')$, where $\mathsf{tr} = ((pk_1, \mathsf{e}_1), \ldots, (pk_{v'(n)}, \mathsf{e}_{v'(n)}))$.
- For each $j \in [v'(n)]$, set $\mathcal{Q}^j_{\mathsf{Eve}} = \mathsf{Adv}^c_1(pk_j)$.
- Ouput $(c, y, c^i, y^i, i, \mathsf{st}, \widetilde{\mathsf{tr}})$, where $\widetilde{\mathsf{tr}} = ((pk_1, \mathcal{Q}^1_{\mathsf{Eve}}, \mathsf{e}_1), \ldots, (pk_{v'(n)}, \mathcal{Q}^{v'(n)}_{\mathsf{Eve}}, \mathsf{e}_{v'(n)}))$.

Fig. 4: The distribution $\widetilde{\mathcal{T}}$.

We additionally define the distribution $\mathcal{T}^{i^*}$ (resp. $\widetilde{\mathcal{T}}^{i^*}$) for $i^* \in [k]$ as the distribution $\mathcal{T}$ (resp. $\widetilde{\mathcal{T}}$), conditioned on $i := i^*$, and the distribution $\mathcal{T}(z)$ (resp. $\widetilde{\mathcal{T}}(z)$) for $z \in \mathsf{Range}(f)$ as the distribution $\mathcal{T}$ (resp. $\widetilde{\mathcal{T}}$), conditioned on $y^i := z$. Even when $z \notin \mathsf{Range}(f)$, we still use the notation $\widetilde{\mathcal{T}}(z)$. This refers to a distribution which is sampled with the same sampling algorithm as the one used for $\widetilde{\mathcal{T}}$, except $y^i := z$ is always fixed to a constant value (not necessarily in the range of f). Let $\mathsf{num}_{\widetilde{\mathcal{T}}}$ be the number of random coins to sample from $\mathcal{T}(z)$. Let $\mathsf{N}_{\widetilde{\mathcal{T}}} := 2^{\mathsf{num}_{\widetilde{\mathcal{T}}}}$.

6 The AM Protocol

We begin with a high-level overview of the protocol: Recall that we fix a particular BBN^- reduction $\mathbb{R}$ with parameters $(v(\cdot), t(\cdot))$ and use the particular

adversary Adv with success probability $(\varepsilon_1 = 1 - \delta/4, \varepsilon_2 = \delta)$ defined in Section **??**. Additionally, recall that we denote by $v'(\cdot), t'(\cdot)$ the following polynomials: $v'(n) := v(\max(n, 1/\varepsilon_2(n))$ and $t'(n) := t(\max(n, 1/\varepsilon_2(n)))$ and that we assume WLOG (see discussion in Section **??**) that $\mathbb{R}$ never makes calls to Adv with security parameter $\widetilde{n} > 2 \cdot t'(n) \cdot v'(n)$. On input z, A constructs many (c, y) pairs and runs many copies of the BBN^- reduction $\mathbb{R}^{\mathsf{Adv}}(c, y)$, using Merlin to help simulate the adversary Adv.

Our AM protocol uses the $\mathsf{HidProt}$ and CBC protocols of Akavia et al. [**?**] (see also the full version for more details.) and the $\mathsf{RandSamp}$ protocol (See Lemma **??**) as subroutines. Parameters $\widetilde{\delta} := (\varepsilon')^2/2, \lambda := 1/k^{1/11}$ are both of order $1/\operatorname{poly}(n)$. For $\widetilde{\mathsf{tr}}$ sampled from $\widetilde{\mathcal{T}}(z)_2$, $\mathsf{HidProt}$ will be used to determine the size of the sets $\widetilde{\mathcal{T}}(z)_2^{-1}(\widetilde{\mathsf{tr}})$ and $\mathcal{T}_2^{-1}(\widetilde{\mathsf{tr}})$. For $(pk^w, \mathcal{Q}^w_{\mathsf{Eve}}) \in \widetilde{\mathsf{tr}}$, CBC (along with the non-uniform advice provided to A) will be used to determine the size α of the set $\mathcal{PK}^{-1}(pk^w, \mathcal{Q}^w_{\mathsf{Eve}})$. Given α, $\mathsf{RandSamp}$ will be used to sample preimages from the set $\mathcal{PK}^{-1}(pk^w, \mathcal{Q}^w_{\mathsf{Eve}})$, thus simulating the adversary's (Adv_2's) response. Note that soundness of $\mathsf{HidProt}$ and CBC only hold under specific conditions (see the full version for more details.). Indeed, a key technical part of the proof is showing that the necessary conditions hold. The purpose of the *testing for goodness* subroutine is the following: We show in the analysis that w.h.p when $z \sim f(U_n)$, the distribution $\widetilde{\mathcal{T}}(z)$ is "good," i.e. somewhat close to the distribution $\mathcal{T}$, so the rejection sampling procedure can be employed. On the other hand, if $\widetilde{\mathcal{T}}(z)$ is not "good" (i.e. very far from $\mathcal{T}$), then A can safely output ACCEPT. Our AM protocol is presented in Figure **??**. We next state our main technical result.

Theorem 10 *Assume that there exists a non-adaptive,* BBN^- *reduction* $(E, \mathbb{R})$ *from PKE to* $(1 - \delta/2)$*-weak one way function. Then for any efficiently computable, length-preserving, (nearly) regular function* f*, the above non-uniform AM protocol* Π_f *has completeness* $1 - \varepsilon'$ *and soundness* $1 - \varepsilon'$ *(for small constant* ε'*), for the distributional language* $\overline{\mathsf{Range}(f)}$*, where soundness holds with probability* $1 - 7\delta$ *over* $z \sim f(U_n)$*.*

We note that if f is not length-preserving, it can be made length-preserving, while (nearly) preserving regularity, via the transformation described in Lemma **??**.

To rule out non-adaptive, BBN^- reductions from PKE to one way function, recall that there is a non-adaptive, black-box reduction from OWF to $(1 - \delta/2)$-weak OWF, where the parameters of the reduction depend only on the input-output size and on δ. but not on the description size of the function. Therefore, if there exists a non-adaptive BBN^- reduction $(E, \mathbb{R})$ from PKE to OWF, then for every polynomial q there also exists a non-adaptive, BBN^- reduction $(E, \mathbb{R})$ from PKE to $(1 - \delta/2)$-weak OWF, where $\delta = 1/7q$. By Theorem **??** (and the extension to non-length-preserving f discussed above) this means that that for every efficiently computable, regular function f and every polynomial q, there exists a non-uniform AM protocol for proving $z \notin \mathsf{Range}(f)$, where soundness holds with probability $1 - 7\delta = 1 - 1/q$ over $z \in f(U_n)$. This contradicts our

assumption that there exists an efficiently computable, (nearly) regular function f such that $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\operatorname{poly}}$.

Theorem 11 *Under the assumption that there exists an efficiently computable, regular function f such that $\overline{\mathsf{Range}(f)} \notin \mathsf{Heur}_{1/\operatorname{poly}}\mathsf{AM}^{\operatorname{poly}}$, there is no non-adaptive, BBN^{-} reduction from PKE to one way function.*

It remains to prove Theorem **??**, which we defer to the full version.

7 Acknowledgements

We thank Tal Malkin for insightful discussions on the notion of BBN^{-} reductions and the anonymous reviewers for TCC B-2016 for their many helpful comments.

The AM Protocol Π_f

Common Input: $z \in \{0,1\}^n$. M is proving to A that $z \notin \mathsf{Range}(f)$.
Non-Uniform Advice: The values $[\mu_\ell^w]_{w\in[v'(n)],\ell\in[p_2(n)/\widetilde{\delta}]}$, where $\mu_\ell^w = \mathrm{E}_{\mathcal{T}_2}[\lfloor(\log_{1+\lambda}\frac{|\mathcal{PK}^{-1}(PK^w,\mathcal{Q}^w_{\mathsf{Eve}})|}{1+\ell\frac{\widetilde{\delta}\lambda}{p_2(n)}})\rfloor]$, and $(PK^w, \mathcal{Q}^w_{\mathsf{Eve}})$ is the random variable corresponding to the w-th pair in $\widetilde{\mathsf{TR}}$.

1. $\mathsf{A} \leftrightarrow \mathsf{M}$**:**
   (a) Let $M := 2k^8$ and let $p_1(n) = p_2(n)\cdot(p^*(n)+1)$. Set $\mathsf{ctr}, \mathsf{ctr}_1 = 0$.
   (b) **Testing for Goodness:**
      i. A samples $\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{p_1(n)} \sim \widetilde{\mathcal{T}}(z)_2$.
      ii. A, M run $\mathsf{HidProt}^{p_1(n)}(\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{p_1(n)})$, returning sizes $[(\rho_i, \tau_i)]_{i\in p_1(n)}$.
      iii. For $j \in [p_1(n)]$, A does the following local computation:
         A. Set $\zeta_i' := \rho_i/\mathsf{N}_{\mathcal{T}}, \gamma_i' := \tau_i/\mathsf{N}_{\widetilde{\mathcal{T}}}$ (approximating $\Pr_{\mathcal{T}_2}[\widetilde{\mathsf{tr}}]$, $\Pr_{\widetilde{\mathcal{T}}(z)_2}[\widetilde{\mathsf{tr}}]$).
         B. If the number of $i \in [p_1(n)]$ such that $\zeta_i' > 2M/3\cdot\gamma_i'$ or $\gamma_i' > 2M/3\cdot\zeta_i'$ is at least $\frac{3p_1(n)}{2k^{1/6}}$, A outputs ACCEPT.
   (c) **Rejection Sampling:**
      i. A samples $[(c_j, y_j, c_j^i, y_j^i, i_j, \mathsf{st}_j, \widetilde{\mathsf{tr}}_j)]_{j\in p_1(n)} \sim \widetilde{\mathcal{T}}(z)$.
      ii. A, M run $\mathsf{HidProt}^{p_1(n)}(\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{p_1(n)})$, returning set sizes $[(\rho_j, \tau_j)]_{j\in[p_1(n)]}$. For $j \in [p_1(n)]$, set $\zeta_j' := \rho_j/\mathsf{N}_{\mathcal{T}}, \gamma_j' := \tau_j/\mathsf{N}_{\widetilde{\mathcal{T}}}$.
      iii. A sets $j = 0$ and repeats the following local computation until $j = p_1(n)+1$:
         A. Set $j = j+1, \mathsf{ctr}_1 = \mathsf{ctr}_1 + 1$.
         B. If $\zeta_j'/M\gamma_j' \le 1$, sample $u \sim U_{[0,1]}$ and check whether $u < \zeta_j'/M\gamma_j'$.
         C. If yes, set $\mathsf{ctr}_1 = 0$, $\mathsf{ctr} := \mathsf{ctr}+1$, $c_{\mathsf{ctr}} := c$, $\widetilde{\mathsf{tr}}_{\mathsf{ctr}} := \widetilde{\mathsf{tr}}_j$ and $j = \mathsf{ctr}\cdot(p^*(n)+1)$, where $\widetilde{\mathsf{tr}}_{\mathsf{ctr}} := [(pk^w_{\mathsf{ctr}}, \mathcal{Q}^w_{\mathsf{Eve},\mathsf{ctr}}, \mathsf{e}^w_{\mathsf{ctr}})]_{w\in[v'(n)]}$.
         D. If $\mathsf{ctr}_1 = p^*(n)+1$, set $\mathsf{ctr}_1 = 0$, $\mathsf{ctr} := \mathsf{ctr}+1$, $c_{\mathsf{ctr}} := \perp$, $\widetilde{\mathsf{tr}}_{\mathsf{ctr}} := \perp$.
   (d) **Simulating Adv_2:**
      i. A, M run $\mathsf{CBC}([pk_j^w, \mathcal{Q}^w_{\mathsf{Eve},j}]_{j\in[p_2(n)],w\in[v'(n)]})$, using non-uniform advice $[\mu_\ell^w]_{w\in[v'(n)],\ell\in[p_2(n)/\widetilde{\delta}]}$, returning set sizes $[\alpha_j^w]_{j\in[p_2(n)],w\in[v'(n)]}$.
      ii. A, M simulate $\mathsf{Adv}_2(pk_j^w, \mathcal{Q}^w_{\mathsf{Eve},j})$ by running $p_2(n)$ parallel copies of $\mathsf{RandSamp}^{p'(n)\cdot v'(n)}([(pk_j^w, \mathcal{Q}^w_{\mathsf{Eve},j}, \alpha_j^w)^{p'(n)}]_{j\in[p_2(n)],w\in[v'(n)]})$, returning values $[\mathcal{I}^w_{j,m}, r^w_{j,m}]_{m\in p'(n),j\in[p_2(n)],w\in[v'(n)]}$.

2. **Generating A's Output:** A does the following local computation:
   (a) For $1 \le j \le p_2(n)$, $1 \le w \le v'(n)$, compute $d_j^w := \mathsf{Adv}_3^{c_j}([\mathcal{I}^w_{j,m}, r^w_{j,m}, \mathsf{e}^w_{j,m}]_{m\in p'(n)})$.
   (b) For each $1 \le j \le p_2(n)$, run $\mathbb{R}_2(\mathsf{st}_j, d_j^1, \ldots, d_j^{v'(n)})$, returning outputs $x_1, \ldots, x_{p_2(n)}$.
   (c) Evaluate each $c_j(x_j)$. If during the evaluation some x^* such that $f(x^*) = z$ is queried to f, output REJECT. Otherwise, output ACCEPT.

Fig. 5: AM protocol for proving that z is not in the image of f.