Universal Reductions: Reductions Relative to Stateful Oracles

Benjamin Chan[*], Cody Freitag[†], Rafael Pass[‡§]

September 19, 2022

Abstract

We define a framework for analyzing the security of cryptographic protocols that makes minimal assumptions about what a "realistic model of computation is". In particular, whereas classical models assume that the attacker is a (perhaps non-uniform) probabilistic polynomial-time algorithm, and more recent definitional approaches also consider quantum polynomial-time algorithms, we consider an approach that is more agnostic to what computational model is physically realizable.

Our notion of *universal reductions* models attackers as PPT algorithms having access to some arbitrary unbounded *stateful* Nature that cannot be rewound or restarted when queried multiple times. We also consider a more relaxed notion of *universal reductions w.r.t. time-evolving, k-window, Natures* that makes restrictions on Nature—roughly speaking, Nature's behavior may depend on number of messages it has received and the content of the last $k(\lambda)$-messages (but not on "older" messages).

We present both impossibility results and general feasibility results for our notions, indicating to what extent the extended Church-Turing hypotheses are needed for a well-founded theory of Cryptography.

[*]Cornell Tech, byc@cs.cornell.edu

[†]Cornell Tech, cfreitag@cs.cornell.edu. This work was done partially during an internship at NTT Research. Supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–2139899.

[‡]Cornell Tech and Tel-Aviv University, rafael@cs.cornell.edu. Supported in part by NSF CNS-2149305, NSF Award SATC-1704788, NSF Award RI-1703846, CNS-2128519, AFOSR Award FA9550-18-1-0267, and a JP Morgan Faculty Award. This material is based upon work supported by DARPA under Agreement No. HR00110C0086.

[§]Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, the United States Government, or DARPA.

Contents

1 Introduction

Modern Cryptography relies on the principle that cryptographic schemes are proven secure based on mathematically precise assumptions; these can be general—such as the existence of one-way functions—or specific—such as the hardness of factoring products of large primes. The security proof is a *reduction* that "transforms" any attacker A of a scheme (e.g., a pseudorandom generator) into an attacker A' that breaks the underlying assumption (e.g., inverts an alleged one-way function). More formally, cryptographic security of a single primitive or assumption is often defined as an *interactive game* (a.k.a. a *security game*) between a *challenger* C and an *adversary* A. C sends a random challenge (e.g. a product of two large primes) to A, who tries to respond in such a way—potentially over many rounds—to make the challenger accept (e.g. by sending the individual factors). The game is determined by the challenger C and the primitive is said to be secure if no "realistic" adversary can cause the challenger to accept with some specified probability. (In the sequel, we will abuse notation and often identify the security game simply by the challenger C.) A reduction R from a game with challenger C (i.e., a security game C) to one with challenger C' provides a way to use a successful adversary A in the game C to construct a successful adversary A' in the game C'. This study has been extremely successful, and during the past four decades many cryptographic tasks have been put under rigorous treatment and numerous constructions realizing these tasks have been proposed under a number of well-studied complexity-theoretic hardness assumptions.

In this paper, we revisit what it means to transform the alleged attacker A for the scheme into an attacker A' for the underlying assumption. In particular, the standard cryptographic treatment explicitly assumes that the attacker A is a (perhaps non-uniform) probabilistic polynomial-time (PPT) Turing machine. Thus, when using the scheme in the "real-world", the security proof is only meaningful if this model of the attacker *correctly captures the computational capabilities of a real-life attacker*—that is, the PPT model correctly captures all "real-life" computation that can be feasibly carried out by an attacker in our physical world. The *extended Church-Turing hypothesis* stipulates that this is the case:

> **The extended Church-Turing Hypothesis:** *A probabilistic Turing machine can efficiently simulate any realistic model of computation.*

But whether this hypothesis holds is strictly speaking a religious, as opposed to scientific, belief.[1] Indeed, the advent of quantum computing directly challenges this hypothesis. Based on exciting developments in quantum computation, it is becoming increasingly clear that viewing an adversary simply as a polynomial-time Turing machine, or polynomial-size circuit, may not be so "realistic". Quantum computers have access to qubits that we believe cannot be described with classical bits or run by a classical, polynomial-sized circuit. Furthermore, by the no-cloning theorem [WZ82, Die82], quantum states cannot be copied or re-used, which is a common technique used by many classical security proofs. We remark that this impacts the security of protocols/primitives even for security

[1] Without getting too deep into Philosophy, it seems reasonable to argue that the Extended Church-Turing Hypothesis does not pass Popper's falsifiability test [Pop05], as we do not have "shared ways of systematically determining" whether a probabilistic Turing machine cannot perform some task (as testified by the fact that the P v.s. NP problem is still open). As such, the statement of the hypothesis is no different from the classic example of "All men are mortal", which according to Popper's theory is not scientific as we do not have systematic procedures for deducing whether a person is immortal. This is in contrast to assumptions such as "Factoring products of random 1000-bit primes is hard for all physically realizable computation devices", as we do have a systematic way of determining whether some such device manages to complete the task—simply run it.

games where the challenger C is purely classical (i.e., primitives implemented by a classical algorithm that the attacker interacts with using classical communication). In recent years, there has been a successful line of work that has focused on proving the security of cryptographic protocols against quantum adversaries (see e.g. [Sho94, Gro96, AC02, Wat09, BDF+11, Unr12, Zha12, ARU14, Unr16, Mah18, BS20] for examples of cryptographic attacks, constructions, and techniques in a quantum world). A vast set of new cryptographic techniques have been developed to address the idiosyncrasies of quantum computation and their impact on the security of systems. But, to deduce "real-life" security from such security proofs, we still need to rely on a quantum version of the extended Church-Turing hypothesis (stipulating that quantum polynomial-time algorithms/circuits can simulate all realistic models of computation).

This begs the question: could there be *even more powerful, or even just incomparable, realistic adversaries* beyond quantum polynomial-time adversaries? After all, a hundred years ago, modern computers did not exist, and quantum physics was in its infancy. Consequently, predicting the computational power of an adversary a hundred years into the future seems unreasonable. If the quantum extended Church-Turing hypothesis is wrong, because of the advent of a new type of computation, it would force yet another re-examination of cryptography.

In this work, instead of tailoring security reductions to specific classes of increasingly powerful adversaries, we ask:

Can we have a well-founded theory of Cryptography without making assumptions on the limits of "physically realizable models of computation"?

Concretely, what if some human manages to (repeatedly) break the security of some cryptographic scheme. There is currently a heuristic leap of faith in our cryptography treatment that this human (and any physical phenomena they may be using) can be implemented in PPT/QPT. Can this leap of faith be avoided?

In particular, ideally, we would want a theory of cryptography without making any types of extended Church-Turing hypotheses, where the security of some scheme is *only* based on *falsifiable* assumptions of the type that some computational task cannot be solved by a "physically realizable computation".[2]

Towards this goal, we will focus our attention on *classical primitives* (i.e., security games with challengers C that are classical and where the attacker can communicate with C only by using classical communication), but consider attackers with unknown/unbounded computational capabilities. At first sight, doing to seems to inherently require information-theoretic security (and all the standard limitations thereof). But our approach will instead be to consider a *purely-reduction based framework*: Our framework will provide a way to reduce the security of a game with challenger C to one with a challenger C' without assuming anything about the adversary *other than the fact that it continually wins in* C. Now, rather than proving the security of some primitive C w.r.t. PPT attackers based on assumptions of the form "C' cannot be broken by PPT attackers", we will view the reduction from C to C' as the main goal: the existence of such a reduction will then imply the statement *"Security of C with respect to any physically realizable attacker holds as long as security of C' holds with respect to any physically realizable attacker"*, without having to impose any restrictions on what the class of "physically realizable attackers" actually is (as long as

[2]For concreteness, and to simplify notation, we will model attackers as Turing machines so technically we are still relying on the (more reasonable) non-extended Church-Turing hypothesis. But we highlight that nothing in our treatment requires doing so and none of our results would change if we instead allowed any, even non-computable, attackers. See Section 3 for more discussion.

they only communicate with the challenger C using classical communication). We note that this reduction-based approach follows intuitions similar to those by Rogaway in his influential "formalizing human ignorance" paper [Rog06], where a purely reduction-based approach is also advocated for (but for a different reason, and where the standard notion of a reduction is employed).

Let us emphasize that whereas our framework is not imposing any upper bounds on the class of feasible computation (hence the name "universal"), we will be assuming a lower bound: in accordance with the standard literature, we will use PPT as a *lower bound* on what can be feasibly done by an attacker.[3] (In other words, polynomial-time computations will be considered realistically feasible, today and forever in the future.) Additionally, we will here focus our attention only on cryptographic primitives where the honest players are standard PPT machines (as opposed to e.g. quantum).

1.1 Universal Reductions in a Nut-shell

Towards defining our reduction-based notion of security, we need to start off by specifying the notion of an attacker we consider. An *augmented adversary* (A, Nat) consists of a uniform PPT interactive Turing machine (ITM) A, known as the *attacker*, and a *stateful*, potentially unbounded ITM Nat, known as the *Nature*. We think of A as the part of the augmented adversary that only uses "standard" computational resources, whereas Nat is a shared resource in the world that may have "magical" computational resources. The stateful nature of Nat is what distinguishes our model from more standard models of "black-box" security used in cryptography. We think of A as some real-life attacker (using today's readily available computing infrastructure) that can interact with a physical Nature Nat. Furthermore, A's interactions with Nature may in turn alter Nature. For instance, if Nat can capture quantum physical phenomena, then by the no-cloning theorem [WZ82, Die82], any type of measurements of Nat may alter it in ways that cannot be reversed (without losing information). Thus, statefulness is key for capturing this.

Roughly speaking, we say that there is a *universal reduction* from a security games C to a game C' if for every PPT A, there exists a "transformed" PPT attacker A' such that for every Nature Nat, if the augmented adversary (A, Nat) wins in the security game C, then the augmented adversary (A', Nat) wins in the security game C'. In other words, the new transformed attacker A' needs to make use of the same Nature Nat as A.[4] (As the reader may notice, this notion captures an "existential" as opposed to a "constructive" notion of a reduction—that is, we are only required to show that a transformed attacker A' *exists*, as opposed to constructively providing it using an efficient transformation from A; we will also discuss constructive notions of reductions below.) We emphasize that A' may only communicate with Nat; it may not rewind, restart, or see any of the implementation details of Nat. In essence, we require A' to win in C' by making use of Nature, much like the original attacker A did, and taking into account that its interaction with the cosmos may alter it. The reasons we model A and A' as PPT, is that we consider PPT as a *lower bound* on what is currently feasible, and assume that this lower bound is valid not only today but also in the future (i.e., we will be able to only do more computation in the future). Thus we can write

[3] This model clearly oversimplifies as, say, n^{100} computation is not actually feasible. But we start off with a standard asymptotic treatment to get a model that is easy to work with. In practice, a more concrete treatment is desirable, but we leave this for future work.

[4] We refer to such reductions as "universal" because they are agnostic to the computational resources of an attacker (and thus can be "universally" applied, independent of the attacker's computational power). Additionally, on a technical level, and as we discuss in more detail shortly, considering security relative to a stateful entity is related to how security is defined in the framework for Universal Composability of Canetti [Can01].

security proofs today that hold regardless of how powerful the universe ends up being (i.e. even if the extended Church-Turing hypothesis turns out to be true). All non-PPT computation can be thought of as being inside Nat.

Comparison with Relativized Reductions and UC security. Before proceeding to further formalizing this notion, let us briefly point out some technical similarities and differences with the notion of a *relativized reduction* (see e.g., [IR95]); roughly speaking, a relativized reduction, and the related notion of a black-box reduction, is a reduction that works even if the attacker has access to some arbitrary (perhaps non-efficienctly computable) function (a.k.a. the "oracle"). The main difference between the notion of a universal reduction and that of relativized reductions is that universal reductions can be viewed as reductions that relativize also with respect to an *interactive, stateful* oracle, whereas relativized reductions are only required to work in the presence of a *non-interactive, stateless*, oracle. As we explained above, considering stateful, interactive, Natures is a crucial aspect of our definition; as we shall see shortly, even formalizing how to deal with stateful oracles/Natures will be non-trivial.

We highlight that the idea to consider cryptographic protocols in the presence of an external stateful entity is also not entirely new: the notion of Universally Composable (UC) security [Can01] does exactly this but in a different context—more specifically, in the context of simulation as opposed to in the context of reductions; see Section 1.7 for more discussion on the relationship between universal reductions and UC.

1.2 Formalizing Universal Reductions

To formalize the notion of a universal reduction we first need to define what it means for (A, Nat) to win in some security game C. The standard notion of winning simply requires the attacker to succeed in convincing C *once* with some probability p. For us, since we consider stateful Natures, this will be too weak. A stateful Nature Nat may decide to be helpful in winning with C just once, and then never again, and such a Nature may not be very helpful in breaking some underlying assumption (at least not repeatedly). In the standard models of reductions, this is not a problem since the attacker can simply be restarted, but this is not allowed in our setting. Consequently, to get a meaningful notion of security, we will restrict our attention to (ruling out) attackers that *repeatedly*, or *"robustly"*, win in the security game, no matter what other communications are taking place with the cosmos. In more detail, we consider any history of interaction ρ that Nat may have seen, where an interaction prefix ρ consists of the messages Nature has received and the random coins it may have tossed. We then require (A, Nat) to succeed in winning for C even if (A, Nat) is fed any such prefix ρ. We denote such an interaction, where entities are provided the security parameter 1^λ, as $\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)$.

In other words, we are considering an attacker A that is interacting with some physical stateful Nature Nat with unknown computational capabilities, but also consider the possibility that there are others in the world (captured by the prefix ρ) that have interacted with Nature in ways that are unknown to the attacker. Still, the attacker needs to succeed in breaking C *no matter what those other prior communications are* (i.e. given any transcript of interactions that previously took place). In fact, this transcript may be of any length, that is, more than just polynomial in λ (noting that Nat may have more than polynomially many interactions in the past).[5]

[5]Nevertheless, we note that all our results also hold if restricting the length of ρ to be polynomial.

Definition 1.1 (Robustly Winning Security Game; Informal)**.** Let C be a challenger in a security game. We say that an *augmented adversary* (A, Nat) *has robust advantage* $a(\cdot)$ *in* C if, for every prefix view ρ, security parameter $\lambda \in \mathbb{N}$, it holds that C outputs 1 with probability at least $a(\lambda)$ in the interaction $\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)$.

Given this notion of robust winning, we can now capture the above-mentioned notion of a universal reduction.

Definition 1.2 (Universal Reduction; Informal)**.** Let C and C' be security games. We say that there is a ϵ*-universal reduction from* C *to* C' if for every PPT A, there exists some PPT A', such that for every Nat, if the augmented adversary (A, Nat) has robust advantage $a(\cdot)$ in C, then (A', Nat) has robust advantage $\epsilon'(\cdot)$ in C' where $\epsilon'(\lambda) = \epsilon(\lambda, a(\lambda))$.

The function ϵ here quantifies the security degradation of the reduction. Let us briefly mention that one may also consider an *a priori* weaker looking notion of a "win-once" universal reduction, that only requires the *transformed* attacker (A', Nat) to have *non-robust* advantage $\epsilon'(\cdot)$ in C'; that is, (A', Nat) is only required to win once in C' as opposed to robustly/repeatedly (while the original attacker (A, Nat) still needs to have robust advantage). As it turns out, this weaker notion is equivalent to the one provided in Definition 1.2; see Lemma 3.2 for more details. We also note that one may consider alternative, seemingly weaker, variants of robustness (e.g., that the attacker only wins an inverse polynomial fraction of the time) but again such a notion turns out to be equivalent (up to a difference in parameters); see Section 3.3 for more details.

Black-box reductions and Dummy adversaries. As mentioned above, the notion of a universal reduction is "existential" as opposed to a "constructive": We do not actually require an efficient transformation taking attackers A to attackers A'; rather, we just need to show that that for every attacker A, the attacker A' *exists*. One could also consider an *a-priori* stronger notion of a *universal black-box reduction* where the transformed attacker A' is defined as $A' = R^A$, where R is fixed PPT (that works for any attacker A). As it turns out, this notion is (again) equivalent to the (existential) notion of a universal reduction provided in Definition 1.2. The reason for this is that to prove the existence of a universal reduction, and actually also a universal black-box reduction, it suffices to show that the reduction applies just to a so-called "dummy" adversary A_{dummy} that essentially just forwards messages between C and Nat; this, intuitively, follows from the fact that we can always push all the work of a prospective attacker A into Nat (more formally, considering a new Nature Nat' that combines Nat and A). We note that a similar phenomena happens for the notion of UC security [Can01], and we are borrowing the term of a "dummy" adversary from there.

Lemma 1.1 (Dummy Lemma; Informal)**.** *Let* C *and* C' *be security games. Assume that there exists some* ϵ *and some PPT* R_{dummy} *such that for every* Nat*, if the augmented adversary* A_{dummy} *has robust advantage* $a(\cdot)$ *in* C*, then* $(R_{\mathsf{dummy}}, \mathsf{Nat})$ *has robust advantage* $\epsilon'(\cdot)$ *in* C' *where* $\epsilon'(\lambda) = \epsilon(\lambda, a(\lambda))$*. Then, there exists an* ϵ-universal black-box reduction from C to C'*.*

We highlight that whereas the actual proof of Lemma 1.1 indeed follows the above intuition, the formalization is quite subtle and quite different from the proof of the dummy lemma in the UC framework—the key obstacle is dealing with the fact that the attacker needs to win robustly.

Note that as a consequence of Lemma 1.1, we have that to prove the existence of a universal reduction, we may without loss of generality assume that $A = A_{\mathsf{dummy}}$ (i.e., in essence that Nat is directly breaking C), and thus proving the existence of a universal reduction amounts to showing the existence of a PPT "filter" $A' = R_{\mathsf{dummy}}$ between C' and Nat.

Composition. We additionally note that the notion of a universal reduction composes. Namely, if hardness of C_1 can be based on the hardness of C_2, and hardness of C_2 can be based on the hardness of C_3, then hardness of C_1 can be based on hardness of C_3.

Theorem 1.2 (Composition Theorem; Informal)**.** *Let C_1, C_2, C_3 be security games. Suppose there exists an ϵ_1-universal reduction from C_2 to C_1, and an ϵ_2-universal reduction from C_3 to C_2. Then, there exists an $\epsilon^\star$-universal reduction from C_3 to C_1 where $\epsilon^\star(\lambda, a) = \epsilon_1(\lambda, \epsilon_2(\lambda, a))$.*

The proof of the composition theorem essentially follows directly from the definition of a universal reduction.

1.3 On the Feasibility of Universal Reductions

We turn to studying the feasibility of universal reductions.

Universal reductions from single-shot, straightline, black-box reductions. We observe that any *straight-line* black-box reduction between C and C' that only invokes the attacker *once* is also a universal reduction. This should not be a surprise since the stateful nature of the attacker in our model never becomes an issue if the reduction only invokes the attacker once. Nevertheless, our model formally demonstrates why such simple types of reductions are advantageous from a (qualitative) security point of view.

Theorem 1.3 (Universal Reductions from Single-shot Straightline Black-box Reductions; Informal)**.** *Let C and C' be security games. Suppose there exists an ϵ-straightline black-box reduction from C to C' that interacts with the adversary once. Then there exists an ϵ-universal reduction from C to C'.*

Fortunately, many well-known reductions in cryptography fall into this class of reductions: PRG length extension, the GGM construction of PRFs from PRGs [GGM86], IND-CPA secure encryption from PRFs, Naor's bit commitments from PRGs [Nao91], and Lamport's one-time signatures from OWFs [Lam79]. We note that for Lamport's construction, this is straightforward to see. For the rest of the proofs, we rely on a uniform security analysis for a hybrid argument, which for example is provided in [Gol07] for PRG length extension. For the convenience of the reader, we provide brief sketches for the constructions and proofs for all of these primitives in Appendix B.

Combining these classical results with Theorem 1.3, we thus directly get the following corollaries (formally stated in Appendix B):

Corollary 1.4 (PRG length extension; Informal)**.** *Let m be a polynomial and G be an $\lambda + 1$-bit stretch PRG. There exists a $m(\lambda)$-bit stretch PRG G_m and an ϵ-universal reduction from the PRG security of G_m to the PRG secutiy of G for $\epsilon(\lambda, a) = 1/2 + \delta/m(\lambda)$, where $\delta = a - 1/2$.*

Corollary 1.5 (PRF from PRGs; Informal)**.** *Let G be any PRG. There exists a PRF F and an ϵ-universal reduction from the PRF secutiy of F to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/\text{poly}(\lambda)$, where $\delta = a - 1/2$.*

Corollary 1.6 (IND-CPA secure private-key encryption from PRGs; Informal)**.** *Let G be any PRG. There exists a private-key encryption scheme and an ϵ-universal reduction from the IND-CPA security of the encryption scheme to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/2 - \mu(\lambda)$ for a negligible function μ, where $\delta = a - 1/2$.*

Corollary 1.7 (Commitment schemes from PRGs; Informal)**.** *Let G be any PRG. There exists a statistically binding commitment scheme and an ϵ-universal reduction from the hiding of the commitment scheme to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/2$, where $\delta = a - 1/2$.*

Corollary 1.8 (One-time Signatures from OWFs; Informal)**.** *Let f be any OWF. There exists a signature scheme and an ϵ-universal reduction from the one-time security of the signature scheme to the OWF security of f for $\epsilon(\lambda, a) = a/(2\lambda)$.*

Universal reductions from new single-shot straightline reductions. Often times, security reductions used in the literature do invoke the attacker multiple times, and it may not be clear how such reductions can be translated to work in the setting of universal reductions. We first show that sometimes famous reductions in the literature that require invoking the attacker multiple times can be made single-shot straightline. In particular, we show that the GMW protocol [GMW91] for graph 3-coloring is witness indistinguishable (WI) [FS90] based on a universal reduction to a commitment scheme (and hence PRGs) with a new proof; the standard proof requires rewinding the attacker and would thus not be applicable in our setting. (This proof may be interesting in its own right; as far as we know, the only proof of WI security of GMW with a straight-line reduction is the work of Hofheinz [Hof11] that shows WI security of GMW when the underlying commitment satisfies a notion of selective-opening security. As far as we know, it was an open problem to present a straight-line reduction based just on standard security; this is what we do.)

Theorem 1.9 (Witness Indistinguishability from PRGs; Informal)**.** *Let G be any PRG. For every language in NP, there exists an interactive proof system (P, V) and an ϵ-universal reduction from the WI of (P, V) to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/\mathrm{poly}(\lambda)$, where $\delta = a - 1/2$.*

Beyond single-shot straightline reductions. While Theorem 1.9 provides some initial hope that more reductions in the literature can be made single-shot straightline, there are other classic reductions that we do not know how to make single-shot. In fact, going one step further, we next show that some classic results in the literature cannot be established with respect to universal reductions.

One of our main results shows that Yao's classic result on hardness amplification of any weak one-way functions via direct product [Yao82] cannot be proven with a universal reduction. In fact, we show that hardness of any arbitrary "black-box" one-way function cannot be amplified essentially *at all* using a n-fold direct product. Given a function f, let $f^{(n)}$ denote the n-fold direct product of f:

Theorem 1.10 (Impossibility of Hardness Amplification; Informal)**.** *Consider some polynomial n, and some function ϵ. Suppose there is an ϵ-universal black-box reduction from the OWF security of $f^{(n)}$ to the OWF security of f that uses only black-box access of f, and that works for any function f. Then, there exists a negligible function μ such that $\epsilon(\lambda, a) \leq a + \mu(\lambda)$.*

Note that there is a trivial reduction that embeds the challenge $f(x)$ a single time into a random location of the output of $f^{(n)}$ that has advantage $\epsilon(\lambda, a) = a$. The above theorem says that no universal reduction can do noticeably better than this trivial reduction, even if considering attackers that succeed with some fixed constant probability, say $\frac{1}{2}$.[6]

[6]We emphasize that Theorem 1.10 is ruling out also so-called "parameter-aware" black-box reductions [BBF13], where the reduction may depend on the success probability a of the attacker; note that Yao's original reduction is

To give some intuition behind the proof of Theorem 1.10, let us recall on a very high-level how Yao's original proof works: given as input y, the reduction embeds y into a random "position" i—letting $y_i = y$, generates random pre-images x_j for $j \neq i$, and lets $y_j = f(x)$, $\vec{y} = y_1 y_2 \dots$ and then runs $A(\vec{y})$. If A fails, then we repeat the process (a polynomial number of times), again embedding y into a new random position i. Note that this reduction is thus repeatedly running A on *correlated* inputs—the inputs all contain the same string y (but except for that y, they are independent). An augmented adversary could notice these correlations and may stop working in case it sees correlations of this form (i.e., a substring y that is repeated from a previous query). Note that such an attacker still robustly wins in the security game: the probability that a fresh input from the challenger coincides with any previously seen strings is negligible.[7] Now, an arbitrary reduction may not necessarily work in the same way as Yao's reduction. However, at a high level, we show that if the reduction works for *any* function f (and only uses the function as a black-box), then the reduction has to ask A on inputs that are correlated, and thus we can still use a similar type of attacker.

We additionally show that the universal aspect of Theorem 1.10 (i.e., that it works for *any* function f) is inherent. If the function f is *rerandomizable* (see Section 4.4 for a formal definition), then we can show a universal reduction for hardness amplification of f—in essence, we show that Yao's reduction directly works. At first sight, this may seem surprising: As mentioned above, Yao's reduction does invoke the attacker multiple time, and does so on correlated inputs (and as discussed above, this correlation lead to problems). Rerandomizability helps overcome this issue and enables the reduction to always feed Nat messages that are independent and have the same distribution.

For our next result, we show that the Goldreich-Levin theorem [GL89] for constructing a OWF with a hardcore predicate from any OWF cannot be turned into a universal reduction, again as long as the underlying OWF is only accessed in a black-box way. For an underlying function g, the Goldreich-Levin theorem shows that the inner product function is hardcore for the "randomized" function $\hat{g}(x, r) = (g(x), r)$. Namely, $\langle x, r \rangle$ cannot be predicted given $(g(x), r)$ where $|x| = |r|$. We extend our impossibility to any predicate h for any length of randomness r (even no randomness).

Theorem 1.11 (Impossibility of a Goldreich-Levin Theorem; Informal)**.** *Consider some function ϵ and some efficiently computable predicate h . Suppose there is an ϵ-universal black-box reduction from the security of the hardcore predicate h w.r.t. $\hat{g}(x, r) = (g(x), r)$ to the OWF security of g that uses only black-box access to g and that works for function g. Then, there is a negligible function μ such that $\epsilon(\lambda, a) \leq \mu(\lambda)$ for all $a \leq 0.99$.*

The proof relies on similar intuitions to the hardness amplifications result.[8] The above theorem gives an indication of why it may be hard to come up with a universal reduction from PRGs to

parameter dependent—more specifically, the number of repetitions is required to be superlinear in the adversary's success probability, and as shown in [LTW05] a dependency on the attackers success probability is inherent for black-box reductions. Theorem 1.10 rules out also such parameter-aware universal reductions and indeed rules out universal reductions that increase the success probability of the adversary even if assuming that the original attackers success probability is, say, $\frac{1}{2}$.

[7] There is a small subtlety here. Robustness is defined with respect to all previous transcripts, even exponentially long ones, so naively implementing this approach will not work since eventually we can include all possible strings y in the transcripts. Rather, the way we formalize this argument is to consider a Nat that only has "polynomial memory" and checks for repeated strings y in the most recent part of the transcript it is fed.

[8] Again, we highlight that Theorem 1.11 rules out also "parameter-aware" reductions that depend on the success probability of the attacker—in fact, it rules out also reductions that only work if the underlying attacker's success probability is 0.99. (As noted in [BBF13], Goldreich-Levin's standard reduction is parameter-aware, and this is inherent as shown in [LTW05].)

OWFs as known constructions of PRGs from OWFs rely on the Goldreich-Levin theorem. We leave open the question of whether there exists some alternative way to universally reduce PRGs to OWFs.

Concluding, while we can write nice universal reductions in some settings, we also have some pretty severe impossibility results. To overcome these impossibility results, we additionally consider more relaxed—yet, in our eyes, natural—variants of universal reductions.

1.4 Restricted Classes of Natures

While it is natural to assume that an attacker can affect Nature/the Cosmos, it also seems reasonable (at least in some contexts) to make additional assumption on the class of Natures. In particular, we will consider Natures that act *independently of the content of interactions they had "far back" in the past*. Roughly speaking, we allow Nature to change over time, and we will allow Nature to be stateful within a single, or a bounded number of, sessions but assume that the actual content of messages received too farin the past (that is, many messages ago) does not significantly affect the behavior of Nature.

In more detail, choose any polynomial function $k(\cdot)$, and consider those natures whose responses depend only on (a) the *number of queries* it has received in the past, (b) the *last* $k(\lambda)$ *messages* that it received, and (c) the randomness that it used to respond to those $k(\lambda)$ messages. We call a nature that satisfies these conditions a *time-evolving* k*-window nature*. We formalize this by requiring that the output of Nature given any two prefixes ρ and ρ' of the *same length* that also share the last $k(\lambda)$ messages and coins, it must be that $\mathsf{Nat}(\rho)$ behaves identically (or ϵ-close to) $\mathsf{Nat}(\rho')$. (The same-length requirement is what allows Nature to evolve over time).

Definition 1.3 (Time-Evolving k-Window Natures)**.** Let $k(\cdot)$ be a polynomial function. A Nature machine Nat is said to be a k-window Nature if there exists a negligible function μ s.t. for all machines C, $\lambda \in \mathbb{N}$, and interaction prefixes ρ, ρ', ρ'', where $\|\rho\| = \|\rho'\|$ and $\|\rho''\| = k(\lambda)$, it holds that

$$\Delta\left(\langle C \leftrightarrow \mathsf{Nat}(\rho \circ \rho'')\rangle(1^\lambda), \langle C \leftrightarrow \mathsf{Nat}(\rho' \circ \rho'')\rangle(1^\lambda)\right) \leq \mu(\lambda).$$

where $\langle C \leftrightarrow S\rangle(1^\lambda)$ denotes the output of C in an interaction with a machine S, Δ denotes statistical distance, $\|\rho\|$ denotes the number of messages contained within ρ, and $\rho \circ \rho''$ denotes prefix concatenation.

Observe that by sending to Nature a sequence of $k(\lambda)$ "dummy messages" $\perp$, we can (roughly speaking) reset the state of a time-evolving k-window Nature, by making it so that its behavior only depends on those dummy messages (and corresponding coins) and the number of messages it received in the past—regardless of the state that Nature started in before receiving those dummy messages. In other words, we can think of an augmented adversary (A, Nat) where Nat is time-evolving k-window (when called repeatedly, each time utilizing the above resetting procedure) as a sequence of attackers $A_1, A_2, \ldots$ such that (1) each individual attacker A_i succeeds in the security game, but (2) the way it succeeds may be different, and (3) the security reduction cannot restart the attacker but may "move on" to the next attacker in the sequence.

As our main result for time-evolving k-window Natures, we show that any *non-adaptive*, straight-line black-box classical reduction can be transformed into a universal reduction, when restricting to time-evolving k-window Natures. In more detail, we refer to a straight-line black-box reduction R as non-adaptive if R interacts with the challenger C and attacker A according to the following pattern:

- R starts by interacting with C for any number of rounds of its choice; at some point it decides that it wants to start communicating with the attacker A.
- At this point, R selects m different PPT machines $M_1, M_2, \ldots, M_m$.
- For each $i \in [m]$, we let M_i communicate (straight-line) with a fresh instance of A, and let a_i denote the output of M_i at the end of the interaction.
- Finally, R gets back the answers $a_1, \ldots, a_m$ and gets to continue interacting with C.

We show:

Theorem 1.12 (Universal Reductions from Non-adaptive Reductions; Informal)**.** *Let C, C' be challengers. If there exists a non-adaptive straight-line black-box reduction from C to C', then for any polynomial $k(\cdot)$, there exists a universal reduction from C to C' w.r.t. time-evolving k-window Natures.*

At a very high level, the idea behind the proof of Theorem 1.12 is the following. Recall that (roughly speaking) an augmented adversary, with a time-evolving k-window Nature, can be treated as a sequence of attackers $A_1, A_2, \ldots$ that is fixed ahead of time and utilized in order. Such a sequence of attackers can essentially be turned into a "standard" fully restartable attacker by, at each invocation, choosing a random attacker A_i out of the sequence of attackers. Of course, in a real execution we are forced to utilize $A_1, A_2, \ldots$ in sequence and in order. Fortunately, for any non-adaptive reduction, we can emulate (with inverse polynomial statistical gap) this standard randomized restartable attacker by *permuting* the order of the queries of the reduction, and inserting these queries into a sufficiently long bogus interaction. Note that we here inherently rely on the fact the a time-evolving k-window Nature can be reset so that the last k messages no longer affects its state, so that its behavior depends on only the length of the prefix of messages it receives.

We remark that many (but not all) of the classical reductions in the cryptographic literature are of the non-adaptive type. In particular, these include reductions such as those in Yao's hardness amplification [Yao82] and the Goldreich-Levin Theorem [GL89] (which we proved could not be shown using a "plain" universal reduction). Perhaps surprisingly, our results therefore imply that we can achieve hardness amplification or hard-core bits for attackers that change their behavior across queries (albeit in this limited way).

k-Window Natures. We finally turn our attention to the more restrictive class of simply *k-window Natures* (i.e., not time-evolving), that are identically defined except that we quantify over any two prefixes ρ and ρ' (with the same last $k(\lambda)$ messages and coins), and not just those of equal length. We observe that straight-line black-box reductions, even those that are *adaptive*, that only sequentially invoke the attacker in multiple sessions, directly imply universal reductions w.r.t. k-window Natures; this essentially follows directly from the definition (by using a standard hybrid argument), and by the observation that sending such a Nature k dummy messages resets it to a default state (from which is acts indistinguishably):

Theorem 1.13 (Universal Reductions from Adaptive Reductions; Informal)**.** *Let C, C' be challengers. If there exists a (possibly adaptive) sequential straight-line black-box reduction from C to C', then for any polynomial $k(\cdot)$, there exists a universal reduction from C to C' w.r.t. k-window Natures.*

1.5 Universal Reductions Imply Standard Reductions

As a sanity check, we finally observe that the existence of a universal reduction from C to C', even one that is only w.r.t. k-window Natures (where $k(\cdot)$ is large enough to bound the number of rounds of interaction with C), implies the existence of a reduction for classic models of attackers such as PPT, non-uniform PPT, quantum polynomial time (QPT), and QPT with non-uniform quantum advice (which we refer to as non-uniform QPT). This follows by noticing that all these models of computations can be captured by a k-window Nature Nat, when used to win a k-round security game C. For the case of PPT, non-uniform PPT, and (uniform) QPT, this is trivial. For non-uniform QPT, it is a bit more problematic since a non-uniform QPT algorithm may make some measurement that ruins the non-uniform advice in a way that makes the algorithm non-restartable. But this issue can be resolved by, for every bound $b(\cdot)$ on the number of restarts, considering a Nat that contains $b(\lambda)$ copies of the non-uniform quantum advice. The resulting attacker (A', Nat) that breaks C' will then still be non-uniform QPT (albeit with longer non-uniform advice than the original attacker breaking C).

Theorem 1.14 (Classical Reductions from Universal Reductions; Informal)**.** *Let C and C' be security games, and let $k(\cdot)$ be a polynomial function that upper bounds the number of rounds in any interaction with C. Assume there exists a ϵ-universal reduction from C to C' w.r.t. (k, μ)-window Natures, for an arbitrary choice of μ. Then there exists a ϵ-reduction from C to C' w.r.t. PPT, non-uniform PPT, QPT, and non-uniform QPT attackers.*

A Note on Post-quantum Security. Note that Theorem 1.14 shows that if you can base the security of some (classical) security game C on the security of C' using a universal reduction (even with respect to just k-window Natures), then it implies resilience of C with respect to quantum attackers if assuming that C' is secure with respect to quantum attackers.

Let us highlight, however, that this result only holds true to security games C that themselves are classical. For instance if C is the security game of a PRF and C' is that of a PRG, then we only get quantum security of the PRF with respect to attackers that can get evaluations of the PRF on classical inputs. (As such, the combination of Corollary 1.5 and Theorem 1.14 does not subsume the results of Zhandry [Zha12] showing post-quantum PRF security of the GGM construction [GGM86] since Zhandry notably allows the attacker to make quantum queries to the PRF.). In other words, our framework currently only consider primitives where the *honest* players are classical. (Of course, we could extend our model to also deal with quantum security games but we believe it is a more pressing issue to get a "future-proof" notion of security w.r.t., cryptographic primitive and protocols that are run by honest players on classical computers).

1.6 Overview of Techniques

We now describe our main technical contributions. We direct the reader to Sections 4 and 5 for full proofs and theorem statements.

1.6.1 The Dummy Lemma

Universal reductions give universal black-box reductions. (See Lemma 1.1) Consider a "dummy attacker" A_{dummy} that forwards all messages from C to the Nature Nat, and forwards replies from Nat back to C. The "dummy lemma" says (informally) that if there exists a universal

reduction R_{dummy} between two security games that works for augmented adversaries of the form $(A_{\mathsf{dummy}}, \mathsf{Nat})$, then there exists a universal reduction that works for any augmented adversary (A, Nat). Moreover, it is constructive, and the resulting reduction uses A in a black-box way. Here, we briefly provide some intuition for why the"dummy lemma" holds.

The key observation is that since R_{dummy} works for any Nature talking with the dummy attacker A_{dummy}, it must in particular also work for the Nature Nat' that internally simulates an attacker A talking to Nat, for any augmented adversary (A, Nat). If (A, Nat) wins some security game C, then $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ should also win an interaction with C, as Nat' is essentially simulating the augmented adversary (A, Nat) inside. Thus, the reduced attacker $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ should also win the game C'. Finally, consider the reduction R^A that internally runs R_{dummy} and forwarding all its attacker messages to its oracle A. Since R_{dummy} is only talking to Nat' in a straightline fashion, intuitively, the augmented adversary (R^A, Nat) should behave exactly like $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ and thus also win C'. Formalizing this intuition, however, is a bit tricky since we need to make sure that $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ is also *robustly* winning in C, which requires a more complicated construction of Nat'; see Section 3.2.

1.6.2 Straightline Black-Box Reductions and Witness Indistinguishability

We overview why single-shot, straightline, black-box reductions imply universal reductions, and use this to show a witness indistinguishable proof based on a universal reduction to PRG security.

Single-shot Straightline Reductions imply Universal Reductions. (See Theorem 1.3) We first argue that "single-shot" straightline black-box reductions imply universal reductions. Suppose there is a *classical* straightline, black-box reduction R that succeeds in some security game C' with probability ϵ when making single-shot usage of an adversary A with advantage a in the game C. That is, R interacts with A a single time without any rewinding or restarting. As we shall observe, any such reduction must also "relativize" with respect to any stateful oracle Nat. In more detail, consider some augmented adversary (B, Nat) that has robust advantage a in a game C, and let B' be an adversary that simulates a communication between R and B: Any time R wants to query its adversary A, we direct that communication to B, and any time B wants to query Nature Nat, we direct that communication to Nat. Since for every prefix ρ, we have that $(B, \mathsf{Nat}(\rho))$ wins in C, we also have that for every prefix ρ, $R^{(B,\mathsf{Nat}(\rho))}$ wins in C and thus $(B', \mathsf{Nat}(\rho))$ (which perfectly emulates $R^{(B,\mathsf{Nat}(\rho))}$) does so as well, so (B', Nat) also has robust advantage in C'. Note that this construction crucially relies on the fact that R only invokes its attacker *once* and without rewinding it (so that communication with Nat can be forwarded).

Let us emphasize, however, that universal reductions are not equivalent with single-shot straightline reductions: as we already discussed, we can obtain universal reductions that do reuse the attacker multiple time—we demonstrate this for the case of hardness amplification for rerandomizable functions—and for this task it is easy to see that a straightline single-shot black-box reductions cannot be used (see Section 4.4 for details).

Universal Reductions from Some Classic Reductions. The above observation shows that if we can construct proofs of security using single-shot, straightline, black-box reductions, then we immediately can infer the existence of a universal reduction. We observe that indeed some of the classical proofs of security (for e.g. PRG length extension, PRFs from PRGs, encryption

from PRFs, commitments from PRGs, one-time signatures from OWFs) fall into this category; see Corollaries 1.4 through 1.8.

Universal Reductions from New Classic Reductions: Witness Indistinguishable Proofs. (See Theorem 1.9) Many classic cryptographic reductions, however, do require rewinding/restarting the adversary. Most notable are reductions/simulations for notions of privacy in interactive proofs like zero-knowledge [GMR89]. As we shall see, we demonstrate that sometimes these can be "de-rewinded". In particular, we will focus our attention on a weakening of zero-knowledge, known as *witness indistinguishability* [FS90], and will show how to provide a new single-shot straighline reduction (and as a consequence, a universal reduction) to PRGs. (We hope that this proof will serve as an example of how classic proofs may be "de-rewinded".)

Recall that an interactive proof system [GMR89], (P, V), for an NP language L specifies an interaction between the prover P with access to a witness w and the verifier V, on common input a security parameter 1^λ and a statement x. It should satisfy completeness, meaning on inputs $x \in L$ and w a valid witness for x, $P(w)$ causes V to accept. The other required property is soundness, meaning on input $x \notin L$, no cheating prover $P^\star$ can cause V to accept (with noticeable probability). Sometimes we want additional privacy and security properties for the witness w used. One basic property is witness indistinguishability (WI) [FS90] which requires that no (potentially cheating) verifier $V^\star$ can tell if P is using one witness w_0 or another witness w_1. Note that this might seem like a weak property (e.g., it provides no guarantees for languages with unique witnesses), but it has been shown to be extremely useful for broader cryptographic applications (see e.g. [FS90, DN07, BG08]).

We show that the GMW protocol for graph 3-colorability [GMW91] is WI using a single-shot straightline reduction to PRG security. We note that previous classical proofs showing WI of the GMW protocol first showed that GMW is actually zero-knowledge and then use this to conclude that it also satisfies WI. But this approach requires rewinding the adversary; we shall dispense of this rewinding.

We proceed to recalling the GMW protocol. Let $G = (U, E)$ be the input graph where $U = [n]$. Recall that the prover P in this protocol has access to a valid 3-coloring $w\colon [n] \to [3]$ such that for all $(i, j) \in E$, $w(i) \neq w(j)$. To prove that the graph G is indeed 3-colorable, P samples a random permutation $\pi\colon [3] \to [3]$ and commits to the colors $c_k = \pi(w(k))$ for all $k \in [n]$. V asks to open a random edge $(i, j) \in E$, and P responds with the openings revealing c_i and c_j. V accepts the interaction if $c_i \neq c_j$ and the openings are valid. Completeness of the protocol can be checked straightforwardly. The protocol has statistical soundness $(1 - 1/|E|)$ (meaning the verifier will catch a cheating prover with probability roughly $1/|E|$) by the statistical binding of the commitment, since at least one edge must be colored incorrectly if G is not 3-colorable. We proceed to argue WI by showing that no cheating verifier $V^\star$ can distinguish interactions with $P(w_0)$ or $P(w_1)$ for any two distinct witnesses w_0 and w_1.

To formalize this claim, we model WI as a security game as follows. We allow the adversary A to select a graph G and two valid witnesses w_0 and w_1. The challenger C samples a bit $b \leftarrow \{0, 1\}$ and proceeds to interact as $P(w_b)$ while A acts as the (potentially cheating) verifier V. After the interaction, A outputs a bit $b^\star$ and C outputs 1 (so A wins) iff $b = b^\star$.

Now suppose that there is an adversary A that distinguishes $P(w_0)$ and $P(w_1)$ with probability $1/2 + \delta$ (namely, it outputs 1 on $P(w_1)$ with probability 2δ more than on $P(w_0)$). We construct a straightline, black-box reduction R that uses A to distinguish two commitments to different values. R first receives a graph G and witnesses w_0 and w_1 from the adversary A. Next, R chooses a random edge $(i', j') \in E$ and random distinct colors for these vertices $c_{i'} \neq c_{j'} \in [3]$. R computes

permutations π_0 and π_1 such that $\pi_0(w_0(\cdot))$ and $\pi_1(w_1(\cdot))$ are consistent with the colors $c_{i'}$ and $c_{j'}$. R then sends two sets of messages to a commitment challenger: the first consists of the colors for $\pi_0(w_0(k))$ for all $k \in U \setminus \{i', j'\}$, and the second consists of the colors for $\pi_1(w_1(k))$ for all $k \in U \setminus \{i', j'\}$. R generates commitments for $c_{i'}$ and c_j and then uses the commitments received from the commitment challenger for the other vertices, so R does not know whether it is using w_0 or w_1. A then asks to open a specific edge $(i, j) \in E$, and if (i, j) happens to be (i', j'), R opens the colors $c_{i'}, c_{j'}$. Otherwise, R aborts. If R didn't abort, the interaction is now over and A outputs a guess $b^\star$ for whether the witness was w_0 or w_1. R simply forwards this guess to the commitment challenger.

Note that by definition, R only queries A in a single session and only via black-box access. So, we only need to argue that R succeeds with better than $1/2$ probability assuming that A succeeds with $1/2 + \delta$ probability for some inverse polynomial δ. At a high level, this follows since A's view is identical to a random execution with either $P(w_0)$ or $P(w_1)$, assuming that R does not abort. The key point in arguing this is that any $b \in \{0, 1\}$, for any fixed edge (i', j') and fixed witness w_b, there is a 1-1 mapping between colors $c_{i'}$, $c_{j'}$ and permutations π_b over colors, so picking random colors for $c_{i'}$, $c_{j'}$ and computing the corresponding permutation w.r.t. w_b, is equivalent to picking a random permutation.

Next, since R chose (i', j') randomly and independent of A, the probability that R aborts because $(i', j') \neq (i, j)$ is at most $(1 - 1/|E|)$. So with probability $1/|E|$, A's guess at distinguishing w_0 from w_1 corresponds exactly to whether or not the commitment challenger chose the commitments for $\pi_0(w_0(\cdot))$ or $\pi_1(w_1(\cdot))$. It follows that R succeeds at distinguishing these two cases with probability $1/2 + \delta/|E|$. Further, we can do an additional hybrid over each of the elements in the set to distinguish two individual committed values with probability $1/2 + \delta/(|E| \cdot (|U| - 2))$.

For full details of the above high level argument, we refer the reader to Section 4.2. The main point is that since this new proof is a single-shot, straightline, black-box reduction, it immediately implies a universal reduction from WI to PRG security.

1.6.3 Impossibility of Hardness Amplification and Goldreich-Levin

Impossibility of Universal Hardness Amplification. (See Theorem 1.10) We start by giving an overview for why there is no universal black-box reduction for the proof of hardness amplification with black-box access to the function f. Let f be a one-way function, and define the n-fold direct product function $f^{(n)}$ such that $f^{(n)}(x_1, \ldots, x_n) = (f(x_1), \ldots, f(x_n))$. We show that this construction does not increase the security for generic functions f. Specifically, we consider generic security games C^f and $C^{(n),f}$ for the OWF security of an arbitrary function f and its n-fold product $f^{(n)}$. Suppose there exists a reduction R such that for any f and any augmented adversary (A, Nat) with advantage $a(\lambda)$ at inverting $f^{(n)}$, then the augmented adversary $(R^{(A,f)}, \mathsf{Nat})$ inverts f with advantage $\epsilon(\lambda, a)$. In this overview, we show that if $R^{(A,f)}$ only makes black-box use of the function f via oracle access to f, then it must satisfy $\epsilon(\lambda, a) \leq a + \mu(\lambda)$ for $a = 1/e$ and μ a negligible function.

Our high level approach is as follows. We will construct an augmented adversary (A, Nat) that has robust advantage roughly $1/e$, yet the answers by this attacker can be efficiently simulated in PPT. In more detail, consider some reduction $(R^{(A,f)}, \mathsf{Nat})$ that work for any function f. Such a reduction must also work for a random function $f : \{0, 1\}^\lambda \to \{0, 1\}^{3\lambda}$, and for random functions, we have the advantage that the reduction won't (except with negligible probability) be able to query the attacker on any point in the range of the function unless it has already queries f on the

pre-image. So, it would seem that if we use such a random function, then we can easily emulate a perfect inverter (by simply looking at all the queries made by R to f). There is one main obstacle here: R actually gets some value y in the range of f as input (and its goal is to invert this point), and R could of course embed this y into its queries to (A, Nat). We overcome this issue by considering a particular "random-aborting" attacker (A, Nat) that (1) only inverts a $1 - 1/n$ fraction of all values y', and (2) never agrees to invert the same value y' twice. We can show that such an attacker succeeds in robustly inverting $f^{(n)}$ with probability roughly $1/e$. Intuitively, such an attacker "knows" how to invert f with probability $1 - 1/n$, but as we shall see, since (A, Nat) is stateful and never agrees to invert the same value twice we can show that (A, Nat) can only be used to invert f with probability roughly $1/e$. More precisely, we show how to correctly simulate this attacker with probability $1 - 1/e$ by a PPT simulator S that simply aborting whenever we see a query that contains a component y_i for which we do not know a pre-image (through one of the f queries made by R). Thus, if $(R^{(A,f)}, \mathsf{Nat})$ inverts a random function f with probability $\epsilon(\lambda, a(\lambda))$, it follows that $(R^{(A,f)}, S^f)$ will invert f with probability $\epsilon(\lambda, 1/e) - 1/e - \mathsf{negl}(\lambda)$. Since R, A, and S are efficient, this probability must be bounded by a negligible function, so $\epsilon(\lambda, 1/e) \leq 1/e + \mathsf{negl}(\lambda)$.

Let us proceed to defining the augmented adversary (A, Nat). The augmented adversary (A, Nat) interacts in the OWF security game of $f^{(n)}$, so A receives queries of the form $(y_1, \ldots, y_n)$. A will simply forward these queries to Nat, who responds with either $\perp$ or the correct inverse $(r_1, \ldots, r_n)$, based on the following procedure:

1. For each y_i in the query, if Nat has previously seen a query for y_i in ρ or if y_i is not in the image of f, it sets r_i to be $\perp$.
2. Next, it flips a coin and with probability roughly $1/n$ just sets r_i to be $\perp$
3. If r_i has not been set to $\perp$, Nat sets r_i to be any preimage in $f^{-1}(y_i)$.
4. Finally, if any r_i was set to $\perp$, Nat responds to the entire query with $\perp$. Otherwise, it responds with the inverse $(r_1, \ldots, r_n)$.

We argue that (A, Nat) will invert a *random* challenge $(f(x_1), \ldots, f(x_n))$ with constant probability, *for all possible prefixes* ρ. In particular, a random challenge $(y_1, \ldots, y_n)$ will always have that each y_i is in the image of f. Additionally, no matter what the history is, a random challenge will not collide with any past query with high probability (formally we need to restrict to only looking at the most recent $\lambda^{\log \lambda}$ queries in case ρ has super-polynomial length). So the only reason Nat outputs $\perp$ is if any of its coin flips tell it to set r_i to be $\perp$, but this happens with probability at most $1 - (1 - 1/n)^n \approx 1 - 1/e$. Thus, the augmented adversary (A, Nat) succeeds with probability roughly $1/e$.

We now argue that Nat can be efficiently simulated. The main reason is that because Nat only needs to reply to queries the first time it sees them, we only need to simulate *a single response* for the challenge $y = f(x)$ that the reduction receives. This is much easier than simulating multiple responses that may include y in various ways. Specifically, the simulator S simulates any queries that R makes to either Nat or f, without the use of Nat. Whenever S simulates a query to f, it records the responses before forwarding the reply back to R. To simulate a query $(y_1, \ldots, y_n)$ to Nat, S proceeds exactly as Nat except that it doesn't actually know how to invert f. Namely, it can still reject y_i values it has seen before, and flip a coin to ignore certain inputs. It tries to invert any y_i value it sees by looking at the queries R has made to f, and uses such a value if one exists.

It remains to argue that S diverges from the behavior of Nat with small probability. S diverges whenever R makes a query $(y_1, \ldots, y_n)$ where R has not queried some y_i before, or if y_i has multiple

pre-images. But because f is a random function from λ to 3λ bits, the probability R can guess an element in the image of f without querying it is negligible (other than its input $y = f(x)$, and the probability that f is not injective is negligible). Thus, we only need to deal with when it queries $y = f(x)$ *for the first time*. But Nat outputs $\perp$ in that case with probability $\approx 1 - 1/e$, so S and Nat only diverge with probability roughly $1/e$!

Finally, it follows that if R, given access to Nat, inverts a random $(y_1, \ldots, y_n)$ with probability $1/e + 1/p(\lambda)$ for some polynomial p, then R given access to the simulator S will invert a random f with probability at least $1/p(\lambda)$, which is impossible. So $(R^{(A,f)}, \mathsf{Nat})$ must invert f with probability at most $\epsilon(\lambda, a) \leq a + \mu(\lambda)$ for $a = 1/e$ and some negligible function μ.

For the above proof, we note that we crucially rely on the fact that (A, Nat) is an augmented adversary because it only ever inverts individual y_i values that it has never seen before. Let us also point out that by setting the abort probability more carefully, we can make the proof go through also when are required to construct an attacker that succeeds with much higher probability a (and not just $1/e$). A rigorous proof is in Section 4.3.

Impossibility of a Universal Goldreich-Levin Theorem. (See Theorem 1.11) We briefly discuss the impossibility of a universal reduction for the Goldreich-Levin theorem. The high level idea and proof structure is similar to the impossibility of hardness amplification.

Recall that the Goldreich-Levin theorem shows that, for any one-way function g, the function $f(x, r) = (g(x), r)$ is a one-way function with hardcore predicate $h(x, r) = \langle x, r \rangle$ for $|x| = |r|$. Let us first outline why the security of the hardcore predicate h cannot be based on the OWF security of g via a universal reduction, when the reduction only has oracle access to the function g.

Similar to the above impossibility for hardness amplification, we construct an augmented adversary (A, Nat) with advantage a where Nat can be efficiently simulated by a machine S for a random function $g\colon \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$. Nat only responds to queries of the form $(g(x), r)$ with the value of $h(x, r)$ (with probability roughly a) once per $g(x)$ value. Then, we construct S that simulates Nat (almost) perfectly except on the first query to the challenge $y = g(x)$ from the OWF challenger. However, since the output of Nat is a single bit, S can just guess what Nat would have output! It follows that S will simulate Nat with roughly $1/2$ probability, so if $(R^{(A,f)}, \mathsf{Nat})$ inverts g with probability $\epsilon(\lambda, a)$, then $(R^{(A,f)}, S^f)$ will do so with probability roughly $\epsilon(\lambda, a)/2$. Since R, A, and S are efficient, this implies that $\epsilon(\lambda, a)$ must be negligible.

Note that we did not use anything about $|r|$ or the structure of h in the above overview. In fact, we rule out any hardcore predicate h for constructions $f(x, r) = (g(x), r)$ for any $|r|$ (even no randomness). See Section 4.5.

1.6.4 Universal Reductions for Time-Evolving k-Window Natures, from Classical Non-adaptive Reductions

Let $k(\cdot)$ be any polynomial function. We here argue that if there exists a *non-adaptive*, straightline black-box reduction R from some game C to C', then there exists a universal reduction from C to C' w.r.t. time-evolving k-window Natures (see Theorem 1.12). For now, we focus on the simplified case where C and C' are 1-round games, but we consider a more general definition of a non-adaptive reductions in Section 5.2.

Recall that a straightline black-box reduction is one where the reduction R only makes black-box use of a classical, stateless adversary A. We say that such a reduction is non-adaptive if (for 1-round games) the reduction R after receiving a challenge message in C', generates m queries

$q_1, \ldots, q_m$ for A in the game C, sends them all at once, receives the responses, and then responds to the challenger C'. Suppose there exists such a reduction R that has advantage ϵ in C' after making m non-adaptive queries to a classical adversary A with advantage a in C. Then for any augmented adversary (B, Nat) with robust advantage a, where Nat is additionally a *time-evolving k-window Nature*, we want to construct an augmented adversary (B', Nat) also with advantage close to ϵ. In particular, for any δ, we will construct B' such that (B', Nat) has robust advantage $\epsilon - \delta$. (This B', however, will have larger running time than R^B, where the running time depends on δ.)

As Nat is a time-evolving k-window Nature, we can essentially think of (B, Nat) as specifying ahead of time a sequence of independent, arbitrary algorithms $S_1, S_2, \ldots$ s.t. it uses S_i to respond to the ith query q_i. We achieve this as follows: in order for B' to be able to emulate such a sequence of attackers $S_1, S_2, \ldots$ using only interactive access to Nat, for each query q_i B' will first send k dummy messages to Nat (in essence resetting its state to be independent of the past, depending only on i). Subsequently, to generate a response for q_i, B' will invoke a fresh copy of B, communicate with Nat on behalf of B, send q_i to B, and reply with B's reply. However, this isn't enough, because each $S_i \in S_1, S_2, \ldots$ may respond differently as i increases (albeit each S_i still wins by robust winning). In other words, the augmented adversary changes over time. To apply the classical non-adaptive reduction R, we must somehow use (B, Nat) to emulate a classical adversary that responds to queries repeatedly according to the same distribution, because R might call its oracle multiple times.

Thus, we construct the universal reduction B' as follows. B' receives some challenge from C' and emulates R on this challenge to generate queries $q_1, \ldots, q_m$. B' then generates $m^2/\delta - m$ extra random "dummy" queries, call them $q_{m+1}, \ldots, q_{m^2/\delta}$. It then samples a random permutation $\pi \colon [m^2/\delta] \to [m^2/\delta]$ that it uses to permute the order of all the queries. For each $i \in [m^2/\delta]$, denote $q'_i = q_{\pi(i)}$. B' then uses $S_1, \ldots, S_{m^2/\delta}$ to respond to those queries, using each S_i to generate a response r'_i for q'_i, in order. It then recovers the responses to the original queries by computing $r_i = r'_{\pi(i)}$ for each $i \in [m]$. R' can feed these to R in order to generate a response for the challenger C'. Importantly, B' is able to emulate $S_1, \ldots, S_{m^2/\delta}$ using a single interaction with the stateful Nat, as long as Nat is a time-evolving (k, μ)-window Nature.

At a high level, the reason the universal reduction B' works is that each response r_i is generated using a random S_j for $j \in [m^2/\delta]$. Thus, R's output should be statistically close to the output of R^A where A is a "classical" adversary A that samples a random $j \leftarrow [m^2/\delta]$ and responds with S_j. However, this isn't the case if there are any collisions on the set of m queries that R queries to this classical adversary A—in other words, if some $j \leftarrow [m^2/\delta]$ is chosen twice—but this bad event can be shown to happen only with probability at most δ. It follows that the output of (B', Nat) is at most δ-far from the output of R^A, so if R wins with probability ϵ, then (B', Nat) will win with probability at least $\epsilon - \delta$.

1.7 Conclusions, Related and Future Work

Interpreting our Results. Our results demonstrate both limitations and feasibility of universal reductions—that is, the feasibility of a foundation for cryptographic security without making extended Church-Turing type assumptions about the class of physically-realistic computations. This paper is only a first step—we have not done an extensive survey of all the reductions in the literature, and we have not investigated all primitives out there; notably, we have focused only on the most basic primitives/reductions. We leave an exploration of more advanced primitives, such as zero-knowledge proofs and secure computation for future work.

Taken together, our result provide a new qualitative understanding of how different types of restrictions on black-box reductions result in security w.r.t. stronger classes of attacker. In particular, when restricting our attention to straight-line black-box reductions: (1) reductions that only invoke the attacker once, yield the strongest form of "plain" universal reduction, (2) reductions that are non-adaptive yield universal reductions w.r.t. time-evolving k-window Natures, and (3) adaptive ones yield universal reductions w.r.t. k-window Natures, for any choice of polynomial $k(\cdot)$.

So given our three different notions of security (which we have shown all imply standard notions of security), which one should we aim to achieve? Obviously the strongest form of plain universal reduction is the most desirable as it allows us to argue security while making only minimal "religious" assumptions about the class of physically-feasible computation. Our results demonstrate that indeed this notion is achievable for many constructions of interest (e.g., for primitives proven secure using straight-line black-box reduction that call the attacker once, or for some cases even multiple times when the queries are independent). Our impossibility results, however, also demonstrate important limitations, showing that in some situations, stronger types of "religious" assumptions about the class of feasible computation are required. The class of time-evolving k-window Natures seems like a reasonable midpoint between expressivity of the theory and the assumptions made on the class of physically-feasible computation.

More Justification for Time-Evolving k-Window Natures. Let us briefly comment on the recent and independent of work of Bitansky, Brakerski, and Kalai ([BBK22]), who study the quantum security of non-interactive reductions. Similar to us, they propose a framework to deal with stateful attackers, and show that non-adaptive reductions (with a polynomial solution space, including decisional assumptions) imply post-quantum security with a *uniform* reduction. In more detail, [BBK22] leverages the main result of Chiesa et al. [CMSZ21] that shows how to effectively "rewind" quantum attackers for a restricted class of protocols so that they effectively become time-evolving but otherwise stateless (or rather, bounded memory)—[BBK22] refer to such attackers as persistent solvers. Next, [BBK22] rely on a proof that is very similar to the proof of our Theorem 1.12 to show that non-adaptive black-box straight-line reductions can be applied to such attackers.

Note that our Theorem 1.14 shows that universal reduction w.r.t. not only time-evolving k-window, but also simply k-window Natures (which by Theorem 1.13 are implied by also adaptive straight-line black-box reductions) imply quantum security but it requires using a *non-uniform* reduction. By relying on the results of [CMSZ21], [BBK22] effectively show that universal reductions w.r.t. time-evolving k-window Natures have the advantage that the reduction for quantum security—for *specific* security games—becomes fully uniform. Consequently, we take the works of [CMSZ21, BBK22] as further evidence that restricting attention to universal reductions w.r.t. time-evolving k-window Natures is meaningful.

Comparison to Universal Composition (UC). Let us highlight that some of the intuition behind our definition take inspiration from the framework for Universal Composability (UC) by Canetti [Can01]. In particular, a simulator in the UC framework needs to interact with the attacker in a black-box straight line fashion in the presence of any environment, without the power of rewinding or restarting the environment. Clearly, there are many similarities between the notion of an environment and our notion of Nature. As such, one may be tempted to hope that UC protocols automatically have universal reductions. This intuition is misleading (as demonstrated e.g., by our Theorem 1.10). The reason for this is that whereas the simulator in the UC framework

is required to be straight-line (and the attacker/environment is allowed to be fully stateful), the *security proof/reduction* used to argue that the simulation is "correct" (i.e., indistinguishable from the real execution in the eyes of the environment) may very well use rewinding (and in fact often does). In more detail, standard proofs in the UC framework still assume that the environment is a non-uniform PPT machine to reduce security to some assumption (e.g., one-wayness of a function).

It is also worthwhile to compare universal reductions to UC security with an *unbounded environment* (in analogy with how we consider Natures that are unbounded). While such a notion of UC security indeed also would be "future-proof" in the sense that it does not make any assumptions about computational limits on the class of physically realizable computations, the problem with such a notion is that it only enables information-theoretically secure protocols, whereas our goal here is to develop a computational theory of cryptography that is "future-proof". One could consider defining primitives (e.g., one-way functions, PRGs, signatures) as UC functionalities, and consider whether one functionality can be implemented in a UC way using some other functionality with respect to a computationally-unbounded environment; as far as we are aware, such a method has not previously been advocated for and is in line with what we are doing here. However, we highlight that doing this is non trivial for several reasons: (1) it is non trivial to define standard cryptographic primitives as UC functionalities (e.g., how does one define an idealized one-way function); (2) such a treatment would require presenting a *straight-line reduction* that is required to work even if the environment (i.e., Nature in our langauge) only helps the attacker to succeed *once*; as we have argued above, such a notion is overly strong (and it is trivial to present impossibility results for it). In contrast, by focusing directly on a reduction-based framework, we can (1) define primitives in the standard game-based way, (2) only require the reduction to work for attackers that win *robustly* (i.e., repeatedly) to rule out trivial cases when Nature helps the attacker to win just a single time.

Let us finally mention that a natural way to define protocol security in a both universally-composable and universally-reducible way would be to consider the standard UC definition of security, but requiring that the security reductions used to prove indistinguishability of the simulation are universal. We leave an exploration of such protocols for future work.

Comparison to Abstract Cryptography. We end by noting that the frameworks for *abstract cryptography* [MR11], and *constructive cryptography* [Mau11], among other things also have as a goal of building up a theory of cryptography that is independent of the model of computation used to model an adversary. While these frameworks were used to analyze how to obtain higher-level functionality (e.g., secure channel) from advanced primitives (e.g., secure encryption and MACs) and also used to analyze some building blocks (for instance see [Mau02, MP04, MPR07]), as far as we can tell, they have not been used to understand the underlying most basic building blocks that we study here (e.g., hardness amplification of one-way functions, whether one-way functions have hard-core bits, etc). At a very high-level, the idea is to view security reductions among primitives as simulations of one system in terms of another; these simulations, just as in the UC framework, need to be straight-line, black-box, and only invoke the attacker once. As far as we can tell, consequently, the same two differences as presented w.r.t. UC with an unbounded environment also apply here. Most notably, since we restrict attention to attackers that win repeatedly/robustly, we can obtain feasibility results using reductions that invoke the attacker multiple times (and this is also what makes it significantly more challenging to present impossibility results).

We highlight that also in the constructive cryptography, *computational* simulation has been defined to consider tasks requiring computational assumption, but this is defined by restricting

attention to polynomial-time distinguishers, so such computational definitions still rely on a extended Church-Turing assumption. It would be interesting to extend these works by considering a computational notion of indistinguishability based on universal reductions.

2 Preliminaries

We let $\mathbb{N} = \{1, 2, 3, \ldots\}$ denote the set of natural numbers, and for any $n \in \mathbb{N}$, we use $[n] = \{1, \ldots, n\}$ to denote the set from 1 to n. We denote by $x \leftarrow X$ the process of sampling a value x from a distribution X. For a set $\mathcal{X}$, we use $x \leftarrow \mathcal{X}$ to denote the process of sampling a value x from the uniform distribution over $\mathcal{X}$. We use U_n to denote the uniform distribution over $\{0,1\}^n$.

Throughout, we use $\lambda \in \mathbb{N}$ to denote the security parameter. When we say that an event holds for *sufficiently large* $\lambda \in \mathbb{N}$ we mean that there exists an integer $N \in \mathbb{N}$ such that the event holds for all $\lambda \geq N$. In particular, for any function $f\colon \mathbb{N} \to \mathbb{N}$, the set $O(f)$ consists of all functions g such that there exists a constants c such that $g(\lambda) \leq c \cdot f(\lambda)$ for sufficiently large $\lambda \in \mathbb{N}$. We say that a function $f(\lambda)$ is polynomially-bounded if it is in the set $\lambda^{O(1)} = \mathrm{poly}(\lambda)$. We say that a function $\mu\colon \mathbb{N} \to \mathbb{R}$ is negligible if it is asymptotically smaller than any inverse-polynomial function, so for every constant $c > 0$, $\mu(\lambda) \leq \lambda^{-c}$ for sufficiently large $\lambda \in \mathbb{N}$. In this case, we say $\mu \in \mathsf{negl}(\lambda)$.

We use PPT to denote the acronym *probabilistic, polynomial time*. A uniform algorithm A is a constant-size Turing machine. We say that a function f is efficiently computable if there exists a uniform, polynomial-time algorithm A such that $A(x) = f(x)$ for all $x \in \{0,1\}^\lambda$. A non-uniform algorithm $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$ is a sequence of algorithms for all $\lambda \in \mathbb{N}$, and we assume for simplicity that A_λ always receives 1^λ as its first input. A non-uniform PPT algorithm is one where the description size of A_λ is bounded by a polynomial as a function of λ.

For two ensembles of random variables $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$, we say that $\mathcal{X}$ is computationally indistinguishable from $\mathcal{Y}$, denoted $\mathcal{X} \approx \mathcal{Y}$, if for all non-uniform PPT distinguishers $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, $|\Pr[D_\lambda(X_\lambda) = 1] - \Pr[D_\lambda(Y_\lambda) = 1]| \leq \mu(\lambda)$. We define the statistical distance between distributions X and Y, denoted as $\Delta(X, Y)$, to be the minimum over all unbounded distinguishers D for of the value $|\Pr[D(X) = 1] - \Pr[D(Y) = 1]|$. Then, for any function μ, we say that two ensembles $\mathcal{X}$ and $\mathcal{Y}$ are μ-statistically close if for all $\lambda \in \mathbb{N}$, $\Delta(X_\lambda, Y_\lambda) \leq \mu(\lambda)$. If μ is a negligible function, we say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable, and if $\mu = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are identically distributed.

An interactive Turing machine (ITM) is an algorithm M that receives and sends messages to other ITMs. For two ITMs, A and B, we denote $\langle A(x), B(y)\rangle(z)$ to denote B's output in the interaction between A and B on private inputs x and y, respectively, and on common input z.

3 Defining Universal Reductions

In this section, we formally present the notion of a universal reduction.

3.1 The Definition and Some Consequences

Towards this, let us first recall the standard notion of a *security game*, wherein an ITM *Challenger* C interacts with an ITM *Adversary* A: On common input 1^λ, C interacts with A until C outputs a bit $b \in \{0,1\}$. If $b = 1$, we say that the adversary *wins*, and we say that A has advantage a if C

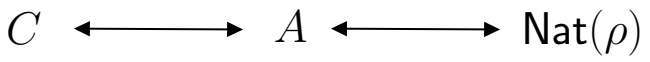

Figure 1: *Execution in a nutshell.* The PPT challenger C plays an interactive security game with a PPT attacker A. To help with generating responses, A may send queries to a potentially unbounded Nature machine Nat. Note that Nat may have had previous interactions, which we specify using ρ, which comprises prior messages that Nat may have received, as well as any private coins that Nat may have flipped previously. When we omit ρ, we mean that Nat starts from the blank slate (i.e. no prior messages or coins).

outputs 1 with probability at least $a(\lambda)$ for all $\lambda \in \mathbb{N}$. The security game is fully specified by the challenger C, and in the sequel we will use security game and challenger interchangeably.

Whereas classically, the adversary is typically a PPT, or a non-uniform PPT, in our context, we will consider security games with respect to *augmented adversaries*: roughly speaking, a PPT attacker A that has access to some potentially unbounded Nature Nat.

Augmented adversaries. In more detail, an augmented adversary (A, Nat) consists of a PPT ITM A, known as the *attacker*, and a stateful, possibly unbounded non-uniform ITM Nat, known as *Nature*. We think of A as the part of the augmented adversary that only uses "standard" computational resources, whereas Nat is a shared resource in the world that may have "magical" computational resources. Note that since Nat is a non-uniform ITM, it may take a non-uniform advice of arbitrary length. We assume that Nat halts on every input message.

Remark 3.1. All of our definitions—and proofs—work for more powerful Natures as well, even those that output an *arbitrary probability distribution* in response to any interaction prefix (as opposed to one being samplable by a TM). We define Nat as an ITM for convenience: It becomes easier to specify communication, randomness, views, etc. Furthermore, considering uncomputable Natures gives incomparable results: the feasibility results are stronger, but the impossibility results become weaker.

Interaction model and winning security games (once). We consider executions of a security game C interacting with an augmented adversary (A, Nat). We use $\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat} \rangle(1^\lambda)$ to denote an execution between C, A, and Nat, given the security parameter 1^λ as common input. In particular, the challenger C sends queries to and receives responses from the attacker A, who in turn sends queries to and receives responses from the Nature machine Nat. The execution ends when C halts outputting a bit $b \in \{0, 1\}$ representing the outcome of the security game. An ITM in this model is PPT if there is a polynomial upper bound—as a function of λ—on the number of steps it takes during the lifetime of any execution before halting. Formally, $\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat} \rangle(1^\lambda)$ is a random variable over the joint views of C, A, Nat, where the randomness is over the coins of each party. Given an execution $\mathsf{exec} \in \mathsf{Supp}(\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat} \rangle(1^\lambda))$, we let $\mathsf{out}_C[\mathsf{exec}]$ and $\mathsf{view}_C[\mathsf{exec}]$ denote C's output and view, respectively, in the execution exec.

Definition 3.1 (Winning Security Games)**.** Let $a \in [0, 1]$ and $\lambda \in \mathbb{N}$ be a security parameter. We say that an augmented adversary (A, Nat) has *advantage a on λ* for a security game C if

$$\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat} \rangle(1^\lambda)] = 1\right] \geq a.$$

Let $a : \mathbb{N} \to [0,1]$. The augmented adversary (A, Nat) has *advantage* $a(\cdot)$ for a security game C if for all security parameters $\lambda \in \mathbb{N}$, (A, Nat) has advantage $a(\lambda)$ for C on λ.

Robust winning. We will also be interested in executions involving Nat where Nat *has already had some prior interaction*; intuitively, we will want to capture a notion of what it means for (A, Nat) to "robustly" win in a security game—roughly speaking, that must (A, Nat) "wins" regardless of any prior interaction that Nat has had with the rest of the world.

We capture this by specifying an *interaction prefix* $\rho = (r, q_1, q_2, \ldots)$ for Nat at the beginning of an execution. We can think of ρ as specifying a finite sequence of queries $q_1, q_2, \ldots$ that Nat previously received, as well as the randomness r that Nat used to respond to those queries; thus ρ fully determines the past behavior and the current state of Nat. For any $\rho \in \{0,1\}^*$ and security parameter $\lambda \in \mathbb{N}$, consider the interaction where Nat is initialized on input 1^λ, with (read-once) random tape prepopulated by r (followed by 0s), and where Nat is reactivated whenever it becomes idle, s.t. when Nat is activated for the ith time, its message tape is prepopulated with q_i (followed by 0s). Recall that an ITM enters an idle state whenever it is ready to receive the next message in the interaction. When there are no more queries in ρ to process, the random tape of Nat is then reset to uniform randomness. We then let $\mathsf{Nat}(1^\lambda, \rho)$ denote Nat in the state reached following the interaction specified by ρ and 1^λ. Let $\|\rho\|$ denote the number of queries sent to Nat in ρ. Finally, the notation $\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)$ refers to an execution on input 1^λ where Nat starts in the state determined by ρ. If the prefix ρ is omitted, then Nat starts without any prior interaction.

We also define what it means to *concatenate* two prefixes $\rho \circ \rho'$, where $\rho = (r, q_1, q_2, \ldots)$ and $\rho' = (r', q_1', q_2', \ldots)$. Define r^* to be the contents of the random tape read by Nat in the interaction $\mathsf{Nat}(1^\lambda, \rho)$, including any 0s if r is too short, or trimming extraneous bits of r that $\mathsf{Nat}(1^\lambda, \rho)$ doesn't read if r is too long. Define $\rho \circ \rho' = (r^* \circ r', q_1, q_2, \ldots, q_1', q_2', \ldots)$, where $r^* \circ r'$ denotes string concatenation.

We are now ready to define what it means for an augmented adversary (A, Nat) to *robustly* win in a security game.

Definition 3.2 (Robust Winning)**.** Let $a \in [0,1]$ and $\lambda \in \mathbb{N}$ be a security parameter. We say that an augmented adversary (A, Nat) has *robust advantage* a on λ for a security game C if for all $\rho \in \{0,1\}^*$, $(A, \mathsf{Nat}(\rho))$ has advantage $a(\lambda)$ on λ for C. Let $a : \mathbb{N} \to [0,1]$. The augmented adversary (A, Nat) has *robust advantage* $a(\cdot)$ for a security game C if for all $\lambda \in \mathbb{N}$, (A, Nat) has robust advantage $a(\lambda)$ for C on λ.[9]

Universal reductions. We finally turn to defining the notion of a universal reduction. Roughly speaking, a universal reduction from security games C to C' guarantees that for every augmented adversary (A, Nat) that robustly wins C, there must exist an attacker A' (depending on A only) such that (A', Nat) robustly wins in C' using the same Nature.

Definition 3.3 (Universal Reductions)**.** Let $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. We say that there is an *ϵ-universal reduction* from C to C' if for all PPT A there exists a PPT A' such that for every augmented adversary (A, Nat) with robust advantage $a(\cdot)$ for C, (A', Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'.

[9] In the definition of robust winning above, we require that the augmented adversary win a security game for *every* prefix ρ that Nat may have previously seen, even those containing exponentially many messages. A natural alternative is to consider a notion of robust winning that considers only those prefixes with $\mathrm{poly}(\lambda)$ many messages; indeed our impossibilities and feasibilities can both be made to work in that setting, but at the expense of definitional complexity.

Composability of Universal Reductions. We observe that the definition of a universal reduction easily composes:

Lemma 3.1 (Composition of Universal Reductions)**.** *Let C_1, C_2, C_3 be security games. Suppose there exists an ϵ_1-universal reduction from C_2 to C_1, and an ϵ_2-universal reduction from C_3 to C_2. Then, there exists an $\epsilon^\star$-universal reduction from C_3 to C_1 where $\epsilon^\star(\lambda, a) = \epsilon_1(\lambda, \epsilon_2(\lambda, a))$ for all $\lambda \in \mathbb{N}$ and $a \in [0,1]$.*

Proof. Let (A_3, Nat) be any augmented adversary, and denote $a(\cdot)$ its robust advantage in C_3. Since there is a ϵ_2-universal reduction from C_3 to C_2, then there exists PPT A_2 s.t. (A_2, Nat) has robust advantage $\epsilon_2(\lambda, a(\lambda))$ in C_2 given security parameter λ for all $\lambda \in \mathbb{N}$. Since there is a ϵ_1-universal reduction from C_2 to C_1, then there must exist PPT A_1 s.t. (A_1, Nat) has robust advantage $\epsilon_1(\lambda, \epsilon_2(\lambda, a(\lambda)))$ in C_1 given security parameter λ for all $\lambda \in \mathbb{N}$.

We conclude that there thus exists a $\epsilon^\star$-universal reduction from C_3 to C_1 where $\epsilon^\star(\lambda, a) = \epsilon_1(\lambda, \epsilon_2(\lambda, a))$ for all $\lambda \in \mathbb{N}$ and $a \in [0,1]$. □

Winning Once v.s. Winning Robustly. Note that the notion of a universal reduction requires taking some augmented adversary (A, Nat) that *robustly* wins in C and transforming it into an augmented adversary (A', Nat) that *robustly* wins in C'. This is useful in order to get a trivial proof for the above composition theorem. However, it also seems natural to consider an *a-priori* weaker definition that only requires the transformed attacker (A', Nat) to *win once* in C' (as opposed to winning robustly):

Definition 3.4 (Win-once Universal Reductions)**.** Let $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. We say that there is a *win-once ϵ-universal reduction* from C to C' if for all PPT A there exists a PPT A' such that for every augmented adversary (A, Nat) with robust advantage $a(\cdot)$ for C, (A', Nat) has advantage $\epsilon(\cdot, a(\cdot))$ for C'.

It turns out that this definition actually is equivalent to the original one.

Lemma 3.2. *Let $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. If there exists a* win-once ϵ-universal reduction *from C to C', then there exists a* ϵ-universal reduction *from C to C'.*

To give some intuition on the proof, observe that the starting augmented adversary (A, Nat) wins robustly for C, and thus for any prefix ρ, letting Nat_ρ denote Nat that hardcodes ρ ahead of time, (A, Nat_ρ) also wins robustly for C. Now, applying the win-once universal reduction, which outputs some A', then for all ρ, (A', Nat_ρ) wins (once) for C', and thus (A', Nat) wins robustly for C'. The formalization follows:

Proof. Assume there exists a win-once ϵ-universal reduction from C to C'. Thus, for any A there exists A' s.t. for any (A, Nat) with some robust advantage $a(\cdot)$ for C, then (A', Nat) has advantage $\epsilon(\cdot, a(\cdot))$ for C'. We claim that (A', Nat) also has robust advantage $\epsilon(\cdot, a(\cdot))$ for C':

For any interaction prefix $\rho \in \{0,1\}^*$, denote Nat_ρ the Nature machine that on input 1^λ, simply runs Nat on input 1^λ, but starting in the state where Nat already saw the prefix ρ. We stress that Nat_ρ hardcodes ρ (as a non-uniform advice string). Then, for any prefix ρ, (A, Nat_ρ) itself also has robust advantage $a(\cdot)$ for C; namely, for all $\rho' \in \{0,1\}^*$, for all $\lambda \in \mathbb{N}$

$$\begin{aligned}
&\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}_\rho(\rho')\rangle(1^\lambda)] = 1\right] \\
&= \Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho \circ \rho')\rangle(1^\lambda)] = 1\right] \\
&\geq a(\lambda)
\end{aligned}$$

where $\rho \circ \rho'$ denotes the concatenation of ρ with ρ'. The inequality follows from the robust advantage of (A, Nat) for C, as $\rho \circ \rho'$ itself is a valid prefix. Consequently, by the correctness of the win-once ϵ-universal reduction, for any prefix ρ, (A', Nat_ρ) has advantage $\epsilon(\cdot, a(\cdot))$ for C'. Thus, for any prefix ρ, and any security parameter λ:

$$\begin{aligned}
&\Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}(\rho))\rangle(1^\lambda)] = 1\right] \\
&\quad = \Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}_\rho\rangle(1^\lambda)] = 1\right] \\
&\quad \geq \epsilon(\lambda, a(\lambda))
\end{aligned}$$

concluding the proof. □

3.2 The Dummy Lemma

In this section we shall show that without loss of generality, it will suffice to present a universal reduction that applies only for a single adversary—the so-called "dummy adversary" A_{dummy} that simply forwards messages between C and Nat. The intuition for why it will suffice for the reduction to work w.r.t. just the dummy adversary is that any other adversary A can be "pushed" into a new Nature Nat' (that combines Nat and A), and we can have the dummy adversary forward messages to it. In fact, we will show an even stronger result: if there exist a universal reduction just for the dummy adversary, there in fact also exists a *black-box* universal reduction—that is, the existence of a *universal* PPT oracle algorithm R (a.k.a. the black-box reduction) such that for every adversary A, the transformed adversary $A' = R^A$.

Definition 3.5 (Universal Black-box Reductions)**.** Let $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. We say that there is an *ϵ-universal black-box reduction* from C to C' if there exists a PPT oracle machine R such that for every augmented adversary (A, Nat) with robust advantage $a(\cdot)$ for C, (R^A, Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'.

A direct corollary of this result will thus be that (non-black box) and black box universal reductions (i.e., Definitions 3.3 and 3.5) are equivalent.

To formalize the above discussion, we will require a slightly more complicated dummy adversary that additionally requires A_{dummy} to send a special $\mathsf{startsession}$ message to Nat' before starting to forward C's messages to Nat'. Roughly speaking, the $\mathsf{startsession}$ message enables Nat' to distinguish messages that should be sent to different invocations of A (i.e. if they originated from different instances of the security game). Additionally, A_{dummy} also tags each of its messages q with a special text $(\text{`}\mathsf{att}\text{'}, q)$, in order to inform Nat' (which again, combines A and Nat) that q should be forwarded to an inner copy of A, as opposed to Nat. This is useful because Nat' also has the option of forwarding incoming messages to Nat, in order to fully simulate the world where A and Nat are separate, and thus Nat' needs a way of distinguishing whether a query is intended for A or for Nat.

Definition 3.6 (The Dummy Attacker A_{dummy})**.** The dummy attacker A_{dummy} is an ITM that interacts with a challenger C and Nature Nat, as follows: on initialization, A_{dummy} sends $\mathsf{startsession}$ to Nat. Subsequently, whenever A_{dummy} receives some message q from the challenger, A_{dummy} forwards $(\text{`}\mathsf{att}\text{'}, q)$ to Nat, and whenever it receives back a response r, A_{dummy} forwards it back to C.

We now define universal reductions with respect to the dummy attacker. It turns out that a universal reduction w.r.t. the dummy attacker implies a standard universal reduction (which works for all attackers). Moreover, the new universal reduction makes black-box use of the attacker. We

note that a similar result hols also in the setting of UC security [Can01] where it without loss of generality suffices to consider security with respect to a particular dummy attacker; we emphasize, however, that the details of the proofs are very different (although the high-level intuition is the same).

Definition 3.7. Let $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games, and R be a PPT ITM. We say that R is an *ϵ-universal reduction* from C to C' *with respect to the dummy attacker* if for every Nature Nat s.t. $(A_{\mathsf{dummy}}, \mathsf{Nat})$ has robust advantage $a(\cdot)$ in C, then (R, Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ in C'.

Theorem 3.3 (Dummy Lemma)**.** *Let $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. If there exists an ϵ-universal reduction from C to C' with respect to the dummy attacker, then there exists a ϵ-universal black-box reduction from C to C'.*

Proof. Let $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. Assume there exists an ϵ-universal reduction from C to C' with respect to the dummy attacker. That is, there exists a PPT R_{dummy} s.t. for all Nat, if $a(\cdot)$ denotes $(A_{\mathsf{dummy}}, \mathsf{Nat})$'s robust advantage in C, then $(R_{\mathsf{dummy}}, \mathsf{Nat})$ has robust advantage $\epsilon(\cdot, a(\cdot))$ in C'. Using this R_{dummy}, we would like to construct a PPT R such that for any attacker A, if $a(\cdot)$ denotes (A, Nat)'s robust advantage in C, then (R^A, Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ in C'.

We proceed to defining the black-box reduction R; see Figure 3 for a visual guide. Recall that given an attacker A, R^A will be interacting with Nat and a challenger C':

- $R^A(1^\lambda)$ simulates $R_{\mathsf{dummy}}(1^\lambda)$ and invokes its oracle A on input 1^λ. Whenever A wants to send a message to Nature, R^A forwards the message to Nat, and delivers the corresponding reply to A. Whenever the simulation of R_{dummy} sends a $\mathsf{startsession}$ message (intended for Nature), R^A restarts A on 1^λ. (Recall that A is a standard PPT so the algorithm can be restarted.)
- Whenever the simulation of R_{dummy} sends to Nature a query of the form $(\text{`}\mathsf{att}\text{'}, q)$, R^A forwards q to its oracle A, and forwards the corresponding reply back to R_{dummy}.
- Whenever the simulation of R_{dummy} sends to Nature a query of the form $(\text{`}\mathsf{nat}\text{'}, q)$, R^A forwards q directly to its own Nature Nat, and forwards the corresponding reply back to R_{dummy}.
- Else if R_{dummy} sends a message m that matches neither format, R^A forwards m directly to Nature, and sends Nature's reply back to R_{dummy}.
- On receiving a query q from the challenger C', R^A sends q to R_{dummy}, and replies with R_{dummy}'s reply.

We now show the correctness of R^A, which requires some work. The proof will follow quite directly from two claims that we state below. The first claim show that we can push A into Nat, creating a new Nature Nat', such that $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ behaves just like (A, Nat). The second claim shows that $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ behaves just like (R^A, Nat). We start by stating the two claims, next formalize why the proof of Theorem 3.3 follows as a consequence of the claims, and then turn to proving the claims.

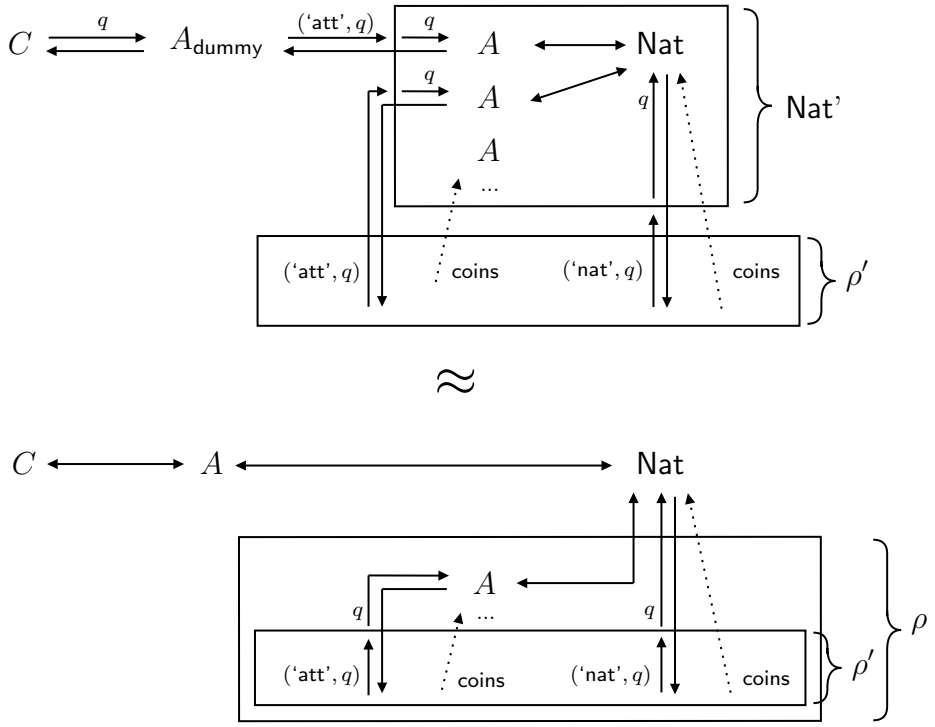

Figure 2: Given an augmented adversary (A, Nat), the composed Nature Nat' (top) internally simulates copies of A each talking with the same Nat in order to respond to queries, creating one copy of A for each instance of security game that Nat' is playing (switching to a new instance of A every time it receives a $\mathsf{startsession}$ message). Note that depending on whether a query q is tagged with 'att' or 'nat', Nat' forwards it either to the active copy of A or to Nat. Any interaction in the $(A_{\mathsf{dummy}}, \mathsf{Nat}'(\rho'))$ world (top) can be simulated perfectly by an interaction in the $(A, \mathsf{Nat}(\rho))$ world for some ρ (bottom). The interaction prefix ρ (for Nat) is constructed using ρ' (for Nat'), stripping messages of their 'att'/'nat' tags, and using the coins specified by ρ' intended for (copies of) A to reconstruct the messages that A sends to Nat.

The Two Central Claims. We state the following two key claims.

Claim 3.4. *Consider some augmented adversary (A, Nat) with robust advantage $a(\cdot)$ for C. Then there exists a Nature Nat' s.t. $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $a(\cdot)$ for C.*

Claim 3.5. *Consider some augmented adversary (A, Nat), and let Nat' denote the combined Nature guaranteed to exists by Claim 3.4. Suppose $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'. Then (R^A, Nat) also has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'.*

Concluding the Proof of Theorem 3.3. Consider some augmented adversary (A, Nat) with robust advantage $a(\cdot)$ in C. By applying Claim 3.4, we have that there exists some Nature Nat' such that $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $a(\cdot)$ for C. By the correctness of the reduction R_{dummy}, it directly follows that $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'. By applying Claim 3.5, it

follows that (R^A, Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ in C', which concludes the proof of Theorem 3.3. □

Proofs of the Claims. In this section, we proceed to proving the two claims used in the proof of Theorem 3.3.

Proof of Claim 3.4. Construct Nat' as follows (see Figure 2 for a visual guide):

- $\mathsf{Nat}'(1^\lambda)$ runs $\mathsf{Nat}(1^\lambda)$ and $A(1^\lambda)$. On receiving a query of the form $(\text{`}\mathsf{nat}\text{'}, q)$, Nat' forwards q directly to Nat, and replies with Nat's reply.
- On receiving a $\mathsf{startsession}$ message, Nat' starts a new fresh simulated copy of $A(1^\lambda)$, replacing any existing copy of A. Denote the newest copy the 'active copy' of A. Throughout, Nat' forwards communication between A and Nat whenever A wants to talk with Nat.
- On receiving a query of the form $(\text{`}\mathsf{att}\text{'}, q)$, Nat' forwards q to the active copy of A, and replies with A's reply.
- On receiving a query m that is not of the above forms, Nat' directly forwards m to Nat, and replies with Nat's reply.

We first argue that $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $a(\cdot)$, as illustrated in Figure 2. Assume for the sake of contradiction that it does not; then there exists an interaction prefix $\rho' \in \{0,1\}^*$ and some λ s.t.

$$\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{dummy}} \leftrightarrow \mathsf{Nat}'(\rho')\rangle(1^\lambda)] = 1\right] < a(\lambda).$$

We would like to use this prefix ρ' to contradict the robust advantage of (A, Nat) for the same λ. For each ρ', define a corresponding prefix for Nat, denoted ρ, constructed as follows:

- Each ρ' specifies an interaction history for Nat, which we denote ρ. This follows because Nat' is internally emulating Nat, so an interaction history for Nat' must also contain a prefix for the emulated Nat. Concretely ρ can be computed from ρ' by stripping messages of their $\text{`}\mathsf{att}\text{'}/\text{`}\mathsf{nat}\text{'}$ tags, using $\mathsf{startsession}$ messages to determine how many copies of A to instantiate, and using the randomness specified in ρ' (intended for simulated copies of A) to reconstruct the messages that A sends to Nat.

Note that

$$\begin{aligned}
&\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)] = 1\right] \\
&\quad = \Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{dummy}} \leftrightarrow \mathsf{Nat}'(\rho')\rangle(1^\lambda)] = 1\right] \\
&\quad < a(\lambda)
\end{aligned}$$

as the simulation is perfect; we are just renaming entities (see Figure 2). We thus contradict (A, Nat)'s $a(\cdot)$-robust advantage for C. Thus $(A_{\mathsf{dummy}}, \mathsf{Nat}')$ has robust advantage $a(\cdot)$ for C. □

We move on to proving Claim 3.5

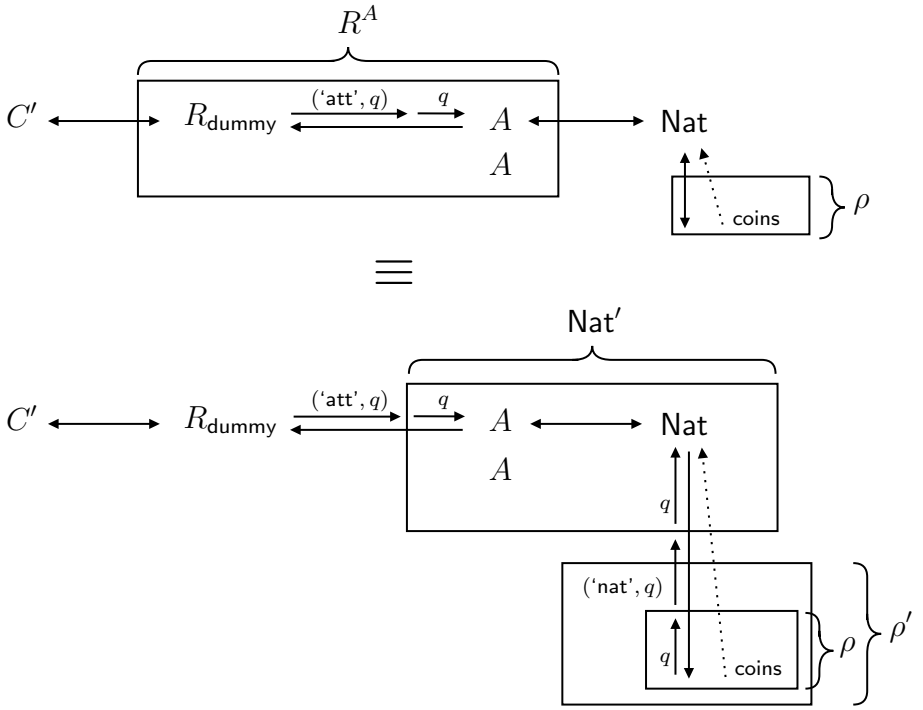

Figure 3: The reduction R^A internally runs a simulation of R_{dummy} and uses its oracle to simulate copies of A in order to generate the correct queries for Nat. Each time R_{dummy} generates a new $\mathsf{startsession}$ message, R^A sends future $(\text{'att'}, q)$ messages to a new copy of A. Here, we show that any interaction with $(R^A, \mathsf{Nat}(\rho))$ (top) can be simulated by an interaction with $(R_{\mathsf{dummy}}, \mathsf{Nat}'(\rho'))$ for some ρ' (bottom). ρ' is essentially the same as ρ, except each incoming message q is additionally tagged with $(\text{'nat'}, q)$.

Proof of Claim 3.5. We show that (R^A, Nat) has the same robust advantage for C' as does $(R_{\mathsf{dummy}}, \mathsf{Nat}')$, by showing that every execution involving (R^A, Nat) can be emulated by an execution with $(R_{\mathsf{dummy}}, \mathsf{Nat}')$. Recall that robust winning requires that for all λ, for every interaction prefix $\rho \in \{0,1\}^*$, $(R^A, \mathsf{Nat}(\rho))$ has advantage $\epsilon(\lambda, a(\lambda))$ for C' on λ. To this end, we show that fixing any λ, for any ρ, there exists ρ' for Nat' s.t.

$$\mathsf{out}_C[\langle C' \leftrightarrow R^A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)] \equiv \mathsf{out}_C[\langle C' \leftrightarrow R_{\mathsf{dummy}} \leftrightarrow \mathsf{Nat}'(\rho')\rangle(1^\lambda)].$$

Given $\rho \in \{0,1\}^*$, construct $\rho' \in \{0,1\}^*$ as follows:

- ρ' is the same as ρ, except each message q in ρ sent to Nature is tagged with $(\text{'nat'}, q)$.

As we shall argue, by construction, this emulation is perfect (see Figure 3 for an illustration). In more detail, the behavior boils down to five cases:

1. R_{dummy} outputs a message $\mathsf{startsession}$. If R_{dummy} is simulated by R^A, R^A restarts its oracle A. If R_{dummy} is interacting with Nat', Nat' will restart its simulation of A.

2. R_{dummy} outputs a message $(\text{'}\mathsf{att}\text{'}, q)$. If R_{dummy} is simulated by R^A, R^A routes q to its oracle A. If R_{dummy} is interacting with Nat', Nat' routes q to its simulated A.

3. R_{dummy} outputs a message $(\text{'}\mathsf{nat}\text{'}, q)$. If R_{dummy} is simulated by R^A, R^A routes q to Nat. If R_{dummy} is interacting with Nat', Nat' routes q directly to its simulated Nat.

4. R_{dummy} outputs any other message m. If R_{dummy} is simulated by R^A, R^A routes m to Nat. If R_{dummy} is interacting with Nat', Nat' routes m directly to its simulated Nat.

Hence,

$$\begin{aligned}
&\Pr[\mathsf{out}_{C'}[\langle C' \leftrightarrow R^A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)] = 1] \\
&\quad = \Pr\big[\mathsf{out}_{C'}[\langle C' \leftrightarrow R_{\mathsf{dummy}} \leftrightarrow \mathsf{Nat}'(\rho')\rangle(1^\lambda)] = 1\big] \\
&\quad \geq \epsilon(\lambda, a(\lambda))
\end{aligned}$$

as required, where the last inequality follows from the $\epsilon(\cdot, a(\cdot))$-robust advantage of $(R_{\mathsf{dummy}}, \mathsf{Nat}')$ for C'. □

Since the existence of a universal reduction trivially implies the existence of a universal reduction for the dummmy adversary, we directly get that universal reductions are equivalent to universal black-box reductions (i.e., Definitions 3.3 and 3.5 are equivalent):

Corollary 3.6. *Let $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. If there exists an ϵ-universal reduction from C to C', then there exists a ϵ-universal black-box reduction from C to C'.*

3.3 An Equivalent Notion of Universal Reduction

We can easily imagine seemingly weaker notions of robust winning that may still admit useful notions of universal reduction. To further convince ourselves that our notion of robust winning is the correct one to study, it is useful to explore whether alternative robust winning definitions are captured by the current notion.

In this section, we consider one such definition. At a high level, consider an augmented adversary that (on input 1^λ and on any prefix ρ) plays a security game C multiple times but is only guaranteed to win 1 out of every $p(\lambda)$ security games that it plays. We show that any universal reduction, given access to such a weak winning augmented adversary, can in fact emulate a robust winning augmented adversary that wins every security game that it plays with probability $1/p(\lambda)$. Thus, showing a universal reduction suffices for this setting as well.

Definition 3.8 (Repeated Security Game)**.** For any security game C, and polynomial $p(\cdot)$, denote $C^{p(\cdot)}_{\mathsf{repeated}}$ the *repeated security game* that on input 1^λ runs $p(\lambda)$ number of copies of C in sequence, starting each subsequent instance of C when the previous copy has halted, and outputting 1 if and only if at least one of its instances of C outputs 1.

Definition 3.9 (Sometimes Robust Advantage)**.** Let $a : \mathbb{N} \to [0,1]$, and let $p(\cdot) : \mathbb{N} \to \mathbb{N}$ be a polynomial function. We say that an augmented adversary (A, Nat) has $p(\cdot)$-*sometimes robust advantage* $a(\cdot)$ for a security game C if (A, Nat) has robust advantage $a(\cdot)$ for $C^{p(\cdot)}_{\mathsf{repeated}}$.

Lemma 3.7 (Equivalence of the two notions)**.** *Let $p(\cdot)$ be a polynomial, and let C be some security game. There exists a PPT oracle machine R s.t. for any augmented adversary (A, Nat), if (A, Nat) has $p(\cdot)$-sometimes robust advantage $a(\cdot)$ for C, then (R^A, Nat) has robust advantage $a(\cdot)/p(\cdot)$ for C.*

Proof. The construction of R is as follows. On input 1^λ, in an interaction with the challenger C and Nature Nat, R^A first samples $i \leftarrow [p(\lambda)]$ uniformly at random. Next, R^A internally runs a single instance of $A(1^\lambda)$ (allowing it to communicate with Nat), and runs in sequence $i-1$ fresh copies of $C(1^\lambda)$ in an interaction with A. After the $(i-1)$th simulation of C (in sequence) has halted, R^A finally starts interacting with its own security game, that is the external C: forwarding C's queries to A, and replying with A's replies, until the external C halts.

Now we claim that (R^A, Nat) has robust advantage $a(\cdot)/p(\cdot)$ for C. Choose any $\lambda \in \mathbb{N}$. Denote $C_1, \ldots, C_{p(\lambda)}$ to be the instances of C run by $C^{p(\cdot)}_{\mathsf{repeated}}$ in any execution on 1^λ, and consider the experiment where we sample $i \leftarrow [p(\lambda)]$ at random, and also independently sample $x \leftarrow \langle C^{p(\cdot)}_{\mathsf{repeated}} \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)$. Then $\Pr[\mathsf{out}_{C_i}[x] = 1]$ over the coins of the experiment is at least $a(\lambda)/p(\lambda)$, since by (A, Nat)'s robust advantage x is a winning execution with probability $a(\lambda)$, and thus one of the instances $C_1, \ldots, C_{p(\lambda)}$ will output 1 by the construction of $C^{p(\cdot)}_{\mathsf{repeated}}$, and this winning instance will be chosen independently with probability $1/p(\lambda)$. Finally, note that $\Pr[\mathsf{out}_{C_i}[x] = 1]$ in the above experiment coincides exactly with $\Pr\big[\mathsf{out}_C[C \leftrightarrow R^A \leftrightarrow \mathsf{Nat}(\rho)] = 1\big]$ by the construction of R, concluding the proof. □

4 Feasibility of Universal Reductions

We next show both positive and negative results for universal reductions. First, in Section 4.1, we show that classical straightline black-box reductions that interact with an adversary in a single session imply universal reductions. This yields many corollaries based on classical reduction, which we provide in Appendix B. In Section 4.2, we provide a new proof for the witness indistinguishability of the GMW protocol [GMW91] from a computationally hiding commitment. We show that this implies a WI proof with a universal reduction to PRG security.

Next, in Section 4.3, we show that hardness amplification via Yao's direct product construction [Yao82] cannot be proven under a universal reduction that makes only black-box use of the underlying one-way function. We show in Section 4.4 that the black-box restriction is necessary since hardness amplification does hold for a notion of *re-randomizable* one-way functions, which we define. In Section 4.5, we show that security of the Goldreich-Levin theorem [GL89] cannot be based on a universal reduction that makes black-box use of the underlying one-way function.

4.1 Straightline Black-Box Reductions

We recall the classical notion of a straightline black-box reduction, where the challenger only interacts in a single session with the adversary. We refer to such a reduction as a *single-shot*, straightline, black-box reduction.

Definition 4.1 (Single-shot Straightline Black-box Reductions)**.** Let C, C' be security games, $a\colon \mathbb{N} \to [0,1]$, $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$. We say that there is an ϵ-single-shot straightline black-box reduction from C to C' if there exists a uniform PPT oracle machine R such that for every

adversary A with advantage $a(\lambda)$ for C on λ, $R^A(1^\lambda)$ has advantage $\epsilon(\lambda, a(\lambda))$ for C' on λ after interacting in a single session with A without rewinding A.

We emphasize here that we require that the oracle machine is *uniform* and cannot hardcode any information that depends on the adversary A. Additionally, we do not restrict ourselves to adversaries A that are efficient. The reduction only makes use of A in a black-box way, independent of its implementation.

We next show that if there is a single-shot straightline black-box reduction R, then that actually does imply a corresponding universal reduction. In Appendix B, we show how this implies universal reductions from a variety of classical constructions and proofs. In particular, we remark that PRG length extension, the GGM PRF construction from PRGs [GGM86], symmetric key encryption from PRFs, Naor's bit commitments from PRGs [Nao91], and Lamport's one-time signatures [Lam79] all fall into this framework.

Theorem 4.1 (Universal Reductions from Straightline Black-box Reductions)**.** *Let C, C' be security games and $\epsilon\colon \mathbb{N}\times[0,1] \to [0,1]$. Suppose there exists an ϵ-single-shot straightline black-box reduction from C to C'. Then there exists an ϵ-universal reduction from C to C'.*

Proof. Let C, C' be security games and $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$. Suppose there exists an ϵ-single-shot straightline black-box reduction R from C to C'. We will show that there exists an ϵ-augmented *win-once* reduction from C to C', which by Lemma 3.2 implies the existence of an ϵ-universal reduction from C to C'. Consider some augmented adversary (B, Nat) that has robust advantage $a(\cdot)$ in the game C; we show that for every λ, $(\tilde{R}^B, \mathsf{Nat})$ has advantage $\epsilon(\lambda, a(\lambda))$ in C' on λ, where $\tilde{R}$ denotes an oracle algorithm that acts just like R but externally forwards (to its own Nature) all Nature queries made by its oracle. Let $(B \leftrightarrow \mathsf{Nat})$ denote the combined execution of B and Nat. Note that by construction, for every $\lambda \in \mathbb{N}$, it holds that

$$\mathsf{view}_{C'}[\langle C' \leftrightarrow \tilde{R}^B \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] \equiv \mathsf{view}_{C'}[\langle C' \leftrightarrow R^{(B\leftrightarrow \mathsf{Nat})}\rangle(1^\lambda)],$$

where $\mathsf{view}_{C'}[\langle C' \leftrightarrow B'\rangle(1^\lambda)]$ is defined as the view of C' in an execution of $C' \leftrightarrow B'$, and thus we have that

$$\begin{aligned}\Pr\Big[\mathsf{out}_{C'}[\langle C' \leftrightarrow \tilde{R}^B \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] = 1\Big] &= \Pr\Big[\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{(B\leftrightarrow \mathsf{Nat})}\rangle(1^\lambda)] = 1\Big] \\ &\geq \epsilon(\lambda, a(\lambda)),\end{aligned}$$

as required. □

4.2 Witness Indistinguishability from PRG Security

The main result of this section is the existence of a proof for any NP language that has a universal reduction from WI security to $(\lambda+1)$-bit stretch PRG security (Theorem 4.3 below). This result follows from a new proof for witness indistinguishability (WI) of the GMW protocol [GMW91] for graph 3-coloring. Our proof is a single-shot straightline black-box reduction from WI to the computational hiding of the underlying commitment, so by Theorem 4.1, it implies a universal reduction as well. For security definitions of PRGs and commitments in our framework, we refer to Appendices B.1 and B.4, respectively.

First, we define witness indistinguishability. Before doing so, recall that an interactive protocol for an NP language L consists of a pair of ITMs (P, V) known as the prover and verifier, respectively.

We use R_L to denote a specific witness relation that defines L, meaning that $x \in L$ iff there exists a witness w such that $(x, w) \in R_L$. In an interactive protocol for L with respect to a witness relation R_L, P and V interact on common input an instance $x \in L$ and security parameter 1^λ, and P additionally receives a private input w such that $(x, w) \in R_L$. We are now ready to define the WI security game for an interactive protocol (P, V).

Definition 4.2 (Witness Indistinguishability)**.** Let (P, V) be an interactive protocol for an NP language L with witness relation R_L. The witness indistinguishability security game is defined as follows. The challenger C interacts with a cheating verifier $V^\star$ on common input 1^λ. First, $V^\star$ sends an instance x and a pair of witness w_0, w_1. The challenger C aborts if either (x, w_0) or (x, w_1) are not in R_L. Otherwise, C samples a bit $b \leftarrow \{0, 1\}$ and emulates P in the interaction $\langle P(w_b), V^\star\rangle(1^\lambda, x)$. Following the interaction, $V^\star$ sends a bit $b^\star$ to C, and C outputs 1 iff $b = b^\star$.

We proceed by recalling the GMW protocol for graph 3-coloring. For simplicity, we assume the existence of a non-interactive, perfectly binding *commitment scheme* Commit for messages in $\{1, 2, 3\}$ that takes as randomness $r \in \{0, 1\}^\lambda$ (which also serves as the opening). However, the proof follows identically for a 2-message scheme, which we formally define and construct in Appendix B.4 by a universal reduction to PRG security (based on standard techniques).

The prover P and verifier V receive as common input a security parameter 1^λ and a graph $G = (U, E)$ where $U = [n]$ is the set of vertices and E is the set of edges. P receives as private input a witness $w \colon U \to [3]$ that specifies a valid three coloring, so for all $(i, j) \in E$, $w(i) \neq w(j)$.

1. P samples a random permutation $\pi \colon [3] \to [3]$. For all $i \in [n]$, P computes $c_i = \pi(w(i)) \in [3]$ as the randomly permuted color of vertex i, samples $r_i \leftarrow \{0, 1\}^\lambda$, and computes $\mathsf{com}_i = \mathsf{Commit}(c_i; r_i)$. P sends com_i to V for all $i \in [n]$.
2. V samples a random edge $(i, j) \in E$ and sends it to P.
3. If P receives an invalid edge, P sets (i, j) to be the first edge in E by default. P sends the openings r_i and r_j to V, revealing the underlying permuted colors c_i and c_j.
4. V accepts if and only if $\mathsf{com}_i = \mathsf{Commit}(c_i; r_i)$, $\mathsf{com}_j = \mathsf{Commit}(c_j; r_j)$, and $c_i \neq c_j$.

We next show that there exists a single-shot, straightline blackbox reduction from the WI of GMW to the computational hiding of Commit.

Lemma 4.2. *There exists an ϵ-single-shot straightline black box reduction from the witness indistinguishability of the GMW protocol to the computational hiding of* Commit*, for $\epsilon(\lambda, a) = 1/2 + \delta/\lambda^3$ where $\delta = a - 1/2$.*

Proof. Consider some attacker A that on input 1^λ outputs a graph $G = (U, E)$ where $|U| = \lambda$, valid witnesses w_0, w_1 for G, and has advantage $1/2 + \delta(\lambda)$ at distinguishing an interaction with $P(w_0)$ from one with $P(w_1)$. (We may assume without loss of generality that A always outputs a valid graph of the right size and valid witnesses, as in case A does not, we can always just pick some dummy graph and valid witnesses.) We construct a PPT reduction R that breaks the hiding of $|U| - 2$ values of Commit given only blackbox access to a single session of A. The theorem statement then follows by an additional hybrid over the $|U| - 2 \leq \lambda$ committed values, which we omit for simplicity. The reduction R is defined as follows.

$R^A(1^\lambda)$:

1. R receives the graph $G = (U, E)$ (such that $|U| = n$) and the valid witnesses w_0, w_1 from A.
2. R next samples a random edge $(i', j') \leftarrow E$ and two distinct colors $c_{i'}$, $c_{j'}$. Let π_0 be a permutation such $c_{i'} = \pi_0(w_0(i'))$ and $c_{j'} = \pi_0(w_0(j'))$, and define π_1 similarly for w_1.
3. R sends the commitment challenger two sets of commitment strings: the first set consists of $\pi_0(w_0(k))$ for all $k \in U \setminus \{i', j'\}$, and the second set consists of $\pi_1(w_1(k))$ for all such k.
4. The challenger samples a random bit $b \leftarrow \{0, 1\}$ and sends R the commitments $\mathsf{com}_k \leftarrow \mathsf{Commit}(\pi_b(w_b(k)))$ for all $k \in U \setminus \{i', j'\}$.
5. R samples $r_{i'}, r_{j'} \leftarrow \{0, 1\}^\lambda$, computes $\mathsf{com}_{i'} = \mathsf{Commit}(c_{i'}; r_{i'})$, $\mathsf{com}_{j'} = \mathsf{Commit}(c_{j'}; r_{j'})$, and sends to the adversary A the strings com_k for all $k \in U$.
6. R receives an edge (i, j) as a response from A. If $(i, j) \neq (i', j')$, R sends the commitment challenger a random bit $b^\star \leftarrow \{0, 1\}$. Otherwise, R sends the openings $r_{i'}$, $r_{j'}$ to A, revealing the underling colors $c_{i'}$, $c_{j'}$. R receives a bit $b^\star$ from A and sends $b^\star$ to the challenger.
7. The commitment challenger outputs 1 if and only if $b^\star = b$.

The following observation will be the central reason this reduction works:

> **Key observation:** *For any fixed edge (i', j') and fixed witness w_b, there is a 1-1 mapping between colors $c_{i'}$, $c_{j'}$ and permutations π_b over colors.*

This simply follows from the fact that in a valid witness w_b, we have that colors on an edge (i', j') must be different, so it suffices determine what those 2 colors get mapped to in order to determine the full permutation (over 3 colors). Note that as a consequence of this observation, it follows that the distribution of coloring obtained by (1) picking a random permutation and applying it to a witness (which is what the honest prover does), is identical to the one obtained by picking two random distinct colors for the edge, computing the unique permutation that corresponds to that coloring with respect to the witness, and then applying this permutation to the witness (which is what happens in the above experiment).

We proceed to analyzing this reduction in more detail. Let us first observe that, as required, R invokes A in a black-box way over a single session and does not rewind or restart A at any point. It remains to analyze the advantage of R^A. Fix some particular security parameter λ, and let $\delta = \delta(\lambda)$. We note that the success probability of R^A is given by

$$\begin{aligned}
\Pr[b^\star = b] &= \Pr[b^\star = 1 \mid b = 1] \Pr[b = 1] + \Pr[b^\star = 1 \mid b = 0] \Pr[b = 0] \\
&= \frac{1}{2} \Pr[b^\star = 1 \mid b = 1] + \frac{1}{2} \Pr[b^\star = 1 \mid b = 0] \\
&= \frac{1}{2} \Pr[b^\star = 1 \mid b = 1] + \frac{1}{2}(1 - \Pr[b^\star = 0 \mid b = 0]) \\
&= \frac{1}{2} + \frac{1}{2} (\Pr[b^\star = 1 \mid b = 1] - \Pr[b^\star = 1 \mid b = 0])
\end{aligned}$$

so it suffices to analyze $(\Pr[b^\star = 1 \mid b = 1] - \Pr[b^\star = 1 \mid b = 0])$.

For a given adversary A and security parameter λ, we define two probabilities for notational convenience. First, for each i, j, we define

$$p_{i,j}(b, i', j', c_{i'}, c_{j'})$$

to be the probability that A outputs the edge (i, j) in response to the first query given R samples i', j' and colors $c_{i'}$, $c_{j'}$ and C samples the bit b. Second, for each i, j, we define

$$q_{i,j}(b, i', j', c_{i'}, c_{j'})$$

to be the probability that A outputs $b^\star = 1$ in response to the second query given that A output (i, j) for its first query, R provides valid openings for vertices i and j, and R and C sampled the given input values as before.

Next, note that by the same argument as we used to expand out the success probability of R^A, it follows that the success probability of A, which by definition is $1/2 + \delta$, can be written as $1/2 + 1/2(W_1 - W_0)$, where W_i is the probability that A outputs 1 after interacting with $P(w_i)$. In other words, we have that

$$W_1 - W_0 = 2\delta$$

Next note by the above "key observation", for every fixed $(i', j') \in E$, we can expand out $W_1 - W_0$ as follows:

$$\begin{aligned} 2\delta &= W_1 - W_0 \\ &= \sum_{c_{i'} \neq c_{j'} \in [3]} (1/6) \cdot \Big(\sum_{(i,j) \in E} p_{i,j}(1, i', j', c_{i'}, c_{j'}) \cdot q_{i,j}(1, i', j', c_{i'}, c_{j'}) \\ &\qquad - p_{i,j}(0, i', j', c_{i'}, c_{j'}) \cdot q_{i,j}(0, i', j', c_{i'}, c_{j'}) \Big). \end{aligned}$$

Pulling out the inner sum and dividing the entire equation by $|E|$, it follows that

$$\begin{aligned} 2\delta/|E| = \sum_{\substack{(i,j) \in E \\ c_{i'} \neq c_{j'} \in [3]}} (1/(6|E|)) \cdot \Big(& p_{i,j}(1, i', j', c_{i'}, c_{j'}) \cdot q_{i,j}(1, i', j', c_{i'}, c_{j'}) \\ & - p_{i,j}(0, i', j', c_{i'}, c_{j'}) \cdot q_{i,j}(0, i', j', c_{i'}, c_{j'}) \Big). \end{aligned}$$

Instead of specifying the colors for the edge (i', j'), we can equivalently go over all options of distinct colors for the edge (i, j), which implies that

$$\begin{aligned} 2\delta/|E| = \sum_{\substack{(i,j) \in E \\ c_i \neq c_j \in [3]}} (1/(6|E|)) \cdot \Big(& p_{i,j}(1, i, j, c_i, c_j) \cdot q_{i,j}(1, i, j, c_i, c_j) \\ & - p_{i,j}(0, i, j, c_i, c_j) \cdot q_{i,j}(0, i, j, c_i, c_j) \Big). \end{aligned}$$

We proceed to analyze $(\Pr[b^\star = 1 \mid b = 1] - \Pr[b^\star = 1 \mid b = 0])$. We start by analyzing $\Pr[b^\star = 1 \mid b = 1]$. $b^\star = 1$ implies that either A output 1 whenever $(i, j) = (i', j')$, or R output a random bit whenever

$(i,j) \neq (i',j')$. Thus, the following holds.

$$\Pr[b^\star = 1 \mid b = 1] = \sum_{\substack{(i,j)\in E \\ c_i \neq c_j \in [3]}} (1/(6|E|)) \cdot p_{i,j}(1,i,j,c_i,c_j) \cdot q_{i,j}(1,i,j,c_i,c_j) + \sum_{\substack{(i,j),(i',j')\in E \\ (i,j)\neq(i',j') \\ c_{i'} \neq c_{j'} \in [3]}} (1/(6|E|)) \cdot p_{i,j}(1,i',j',c_{i'},c_{j'}) \cdot (1/2).$$

A similar equation holds for $\Pr[b^\star = 1 \mid b = 0]$. Thus, writing out the difference, we get

$$\begin{aligned}
&\Pr[b^\star = 1 \mid b = 1] - \Pr[b^\star = 1 \mid b = 0] \\
&= \sum_{\substack{(i,j)\in E \\ c_i \neq c_j \in [3]}} (1/(6|E|)) \cdot \Big(p_{i,j}(1,i,j,c_i,c_j) \cdot q_{i,j}(1,i,j,c_i,c_j) - p_{i,j}(0,i,j,c_i,c_j) \cdot q_{i,j}(0,i,j,c_i,c_j) \Big) \\
&+ \sum_{\substack{(i,j),(i',j')\in E \\ (i,j)\neq(i',j') \\ c_{i'} \neq c_{j'} \in [3]}} (1/(6|E|)) \cdot \Big(p_{i,j}(1,i',j',c_{i'},c_{j'}) \cdot (1/2) - p_{i,j}(0,i',j',c_{i'},c_{j'}) \cdot (1/2) \Big).
\end{aligned}$$

The first term is equal to $2\delta/|E|$ by our analysis above. The second term is equal to 0, seen by the following sequence of equalities.

$$\begin{aligned}
&\sum_{\substack{(i,j),(i',j')\in E \\ (i,j)\neq(i',j') \\ c_{i'} \neq c_{j'} \in [3]}} (1/(6|E|)) \cdot \Big(p_{i,j}(1,i',j',c_{i'},c_{j'}) \cdot (1/2) - p_{i,j}(0,i',j',c_{i'},c_{j'}) \cdot (1/2) \Big) \\
&= (1/(2|E|)) \cdot \left(\Big(1 - \sum_{\substack{(i,j)\in E \\ c_i \neq c_j \in [3]}} (1/6) \cdot p_{i,j}(1,i,j,c_i,c_j) \Big) - \Big(1 - \sum_{\substack{(i,j)\in E \\ c_i \neq c_j \in [3]}} (1/6) \cdot p_{i,j}(0,i,j,c_i,c_j) \Big) \right) \\
&= 0.
\end{aligned}$$

The last line follows since for each $b \in \{0,1\}$ and distinct colors $c \neq c' \in [3]$, it holds that

$$\sum_{(i,j)\in E} p_{i,j}(b,i,j,c,c') = 1.$$

Plugging these values in, it follows that

$$\Pr[b^\star = 1 \mid b = 1] - \Pr[b^\star = 1 \mid b = 0] = 2\delta/|E|.$$

This implies that R has advantage $1/2 + \delta/|E|$ at breaking the hiding of a set of $|U| - 2$ committed values, as required. □

Based on the lemma above, we have the following theorem as an immediate corollary based on standard results and the framework for universal reductions we have built up so far.

Theorem 4.3. *Let g be a $(\lambda+1)$-bit stretch PRG. For any NP language L with witness relation R_L, there exists a constant round interactive protocol (P, V) that satisfies completeness, has negligible statistical soundness, and satisfies the following witness indistinguishability guarantee. There exists an ϵ-universal reduction from the WI security game of (P, V) to the PRG security game of g for $\epsilon(\lambda, a) = 1/2 + \delta/(12\lambda^7)$ where $\delta = a - 1/2$.*

Proof sketch. The prover and verifier first apply a standard NP reduction from the NP language L to graph 3-coloring.

The protocol then consists of repeating the GMW protocol above in parallel $|E| \cdot \lambda$ times. Completeness holds by completeness of the GMW protocol. Soundness holds unconditionally from the statistical binding of the commitment scheme and the standard proof of the GMW protocol. Since a single instance of the protocol has statistical soundness $(1 - 1/|E| - \mathsf{negl}(\lambda))$, the $|E| \cdot \lambda$ repeated protocol has statistical soundness $\mathsf{negl}(\lambda)$.

The proof of witness indistinguishability of the repeated protocol follows by the proof of Lemma 4.2, which incurs a loss of λ^3, by doing another hybrid over the $|E| \cdot \lambda \leq \lambda^3$ repetitions. This gives an advantage of $1/2 + \delta/\lambda^6$ in breaking the hiding of a 2-bit commitment scheme (in order to encode the colors $\{1, 2, 3\}$). We lose an extra factor of 2 in the security by another hybrid to the 1-bit commitment scheme defined in Appendix B.4. The existence of a universal reduction to commitment hiding follows from Theorem 4.1. Finally, the universal reduction from commitment hiding to 3λ-bit stretch PRG security follows from Corollary B.4, which incurs an extra factor of 2 loss, and the composability Lemma 3.1. Finally, from the composability lemma and Corollary B.1, we reduce to a $(\lambda + 1)$-bit stretch PRG while incurring another 3λ loss in security. Thus, overall our advantage in the reduction to the PRG security game is $1/2 + \delta/(12\lambda^7)$ for $\delta = a - 1/2$. □

4.3 Impossibility of Hardness Amplification

We next show that hardness amplification of a one-way function via a direct product cannot be proven with a universal reduction, at least if the transformed attacker only has oracle access to f. In order to formalize the notion of a universal black-box reduction that we rule out, we first define a security game that has oracle access to a specific function. Specifically, we consider security games C that are defined with respect to some arbitrary function $f : \{0,1\}^* \to \{0,1\}^*$ from a class $\mathcal{F}$, and require that the attacker only has oracle access to the function f. To define a security game C that works for any function f, we also give the challenger C oracle access to f. More formally, define a universal black-box reduction with oracle access to a class of functions as follows:

Definition 4.3 (Universal Black-box Reductions with Oracle Access)**.** Let $\mathcal{F}$ be a set of functions $f : \{0,1\}^* \to \{0,1\}^*$. Let $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, and C and C' be PPT oracle machines. We say that there is an *ϵ-universal black-box reduction with oracle access to the primitive $\mathcal{F}$* from C to C' if there exists a PPT oracle machine R s.t. for every $f \in \mathcal{F}$, every augmented adversary (A, Nat), and every $\lambda \in \mathbb{N}$, letting a denote (A, Nat)'s robust advantage for C^f on λ, then $(R^{A,f}, \mathsf{Nat})$ has robust advantage $\epsilon(\lambda, a)$ for C'^f on λ.

We recall the direct product construction first introduced by Yao [Yao82]. Let $f \colon \{0,1\}^* \to \{0,1\}^*$. For any $n \colon \mathbb{N} \to \mathbb{N}$, we define the n-fold direct product of f, denoted by $f^{(n)} \colon (\{0,1\}^\lambda)^{n(\lambda)} \to (\{0,1\}^*)^{n(\lambda)}$, to be the function defined as follows:

$$f^{(n)}(x_1, \ldots, x_{n(\lambda)}) = (f(x_1), \ldots, f(x_{n(\lambda)})).$$

For any polynomial n, and oracle f, we let $C^{(n),f}$ denote the one-way function security game for the direct product construction, that on input 1^λ samples $x \leftarrow \{0,1\}^{\lambda \cdot n(\lambda)}$, uses its oracle access to f to compute $y = f^{(n)}(x)$, sends y to the attacker, and on seeing a reply z, outputs 1 i.f.f. $f^{(n)}(z) = y$ (which $C^{(n),f}$ uses its oracle access to f to check). Note that $C^{(1),f}$ denotes the normal one-way function security game. Note that $C^{(n)}$ and $C^{(1)}$ are both oracle machines, and use f in a black-box way as desired.

Our impossibility result shows that this direct product construction cannot be used to amplify hardness essentially *at all*, even if we only require the reduction to work w.r.t. attackers that succeed with fixed constant probability, say $a = 1/2$: No matter the advantage a of the attacker on the direct product construction $f^{(n)}$, the reduction will only succeed in inverting the underlying function f with roughly the same probability a.

Theorem 4.4 (Impossibility of Black-box Hardness Amplification)**.** *Let $\mathcal{F}$ be the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$. For any polynomial n, suppose there exists an ϵ-universal (black-box) reduction from $C^{(n)}$ to $C^{(1)}$ with oracle access to the primitive $\mathcal{F}$. Then, there exists a negligible function μ such that $\forall \lambda \in \mathbb{N}, a \in [0,1]$, $\epsilon(\lambda, a) \leq a + \mu(\lambda)$.*

Before proceeding to the proof of Theorem 4.4, let us state two (standard) claims about random functions that will be useful for us. For each $\lambda \in \mathbb{N}$, denote $\mathcal{F}_\lambda^{3\lambda}$ the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$ s.t. $|f(x)| = 3|x|$ and for all $x \notin \{0,1\}^\lambda$, $f(x) = x \circ x \circ x$ where $\circ$ denotes concatenation. In plainer English, $\mathcal{F}_\lambda^{3\lambda}$ is the set of all length-tripling functions that on inputs of length *not equal to* λ behave essentially like an identity function (but still length-tripling).

Claim 4.5. *Let $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, then f is injective with probability at least $1 - 2^{-\lambda}$.*

Proof. Every input $x \notin \{0,1\}^\lambda$ already has a unique image $f(x) = x \circ x \circ x$, so consider only those $x \in \{0,1\}^\lambda$. For every input $x \in \{0,1\}^\lambda$, its output is a random value $y \leftarrow \{0,1\}^{3\lambda}$ over a randomly chosen f. Thus, for any $x \in \{0,1\}^\lambda$, the probability that there exists an x' such that $f(x) = f(x')$ is at most $2^\lambda \cdot 2^{-3\lambda} = 2^{-2\lambda}$. By a union bound, it follows that for all $x \in \{0,1\}^\lambda$, the probability there exists an x' such that $f(x) = f(x')$ is at most $2^\lambda \cdot 2^{-2\lambda} \leq 2^{-\lambda}$, which implies the claim. □

Claim 4.6 (essentially in [IR95])**.** *Let $\ell : \mathbb{N} \to \mathbb{N}$ and $\lambda \in \mathbb{N}$. For any oracle attacker B that makes at most $\ell(\lambda)$ queries to its oracle, it holds that:*

$$\Pr\left[\begin{array}{l} f \leftarrow \mathcal{F}_\lambda^{3\lambda} \\ x \leftarrow \{0,1\}^\lambda \\ x' \leftarrow B^f(1^\lambda, f(x)) \end{array} : f(x) = f(x')\right] \leq \frac{3 \cdot \ell(\lambda)}{2^\lambda}.$$

Proof. Let $\ell : \mathbb{N} \to \mathbb{N}$ be any function, fix $\lambda \in \mathbb{N}$, and let B be any oracle attacker that makes at most $\ell(\lambda)$ queries to its oracle. Suppose that B, before outputting some x', always queries its function oracle f on x', without loss of generality. We focus on the probability that $B^f(1^\lambda, f(x))$ queries its function oracle f on x, where the probability is taken over $x \leftarrow \{0,1\}^\lambda$, B's randomness, and $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, conditioned on f being injective.

For each $i \in \mathbb{N}$, denote $\mathsf{UnQueried}_{i-1}$ the event that B did not query x in its first $i-1$ queries, and denote $\mathsf{Queried}_i$ the event that B queries x as the ith query. For each i, $\Pr[\mathsf{Queried}_i \cap \mathsf{UnQueried}_{i-1}] \leq \Pr[\mathsf{Queried}_i \mid \mathsf{UnQueried}_{i-1}]$. In Subclaim 4.7 below, we show that for any choice of $i < 2^\lambda/2$,

$$\Pr[\mathsf{Queried}_i \mid \mathsf{UnQueried}_{i-1}] \leq 2/2^\lambda.$$

B makes at most $\ell = \ell(\lambda)$ queries, so $i \leq \ell$. Suppose that $\ell < 2^\lambda/2$. Then, by a union bound, the overall probability that B queries x at any point is at most $\ell \cdot 2/2^\lambda$. Notice that for $\ell \geq 2^\lambda/2$, then $\ell \cdot 2/2^\lambda \geq 1$ anyways, so the bound extends to all choices of $\ell(\cdot)$.

Finally, by Claim 4.5 $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$ is injective with probability at least $1 - 2^{-\lambda}$; so putting it all together, B outputs x with probability $\leq \ell \cdot 2/2^\lambda + \Pr[\neg \text{ injective}] \leq \ell \cdot 2/2^\lambda + 2^{-\lambda} \leq 3\ell/2^\lambda$.

Subclaim 4.7. *Let* $i < 2^\lambda/2$. *Then* $\Pr[\mathsf{Queried}_i \mid \mathsf{UnQueried}_{i-1}] \leq 2/2^\lambda$.

Proof of Subclaim 4.7. Denote B_i the oracle attacker that runs B until B makes its ith oracle query (or halts), at which point B_i outputs B's ith query (or B's output). Fix any arbitrary input $y \in \{0,1\}^{3\lambda}$, and consider *any* execution $B_i(y)$ – in other words we fix the oracle query/response pairs $(x_1, y_1), \ldots, (x_{i-1}, y_{i-1})$, and finally any corresponding output x'. Moreover, assume that for all $j, k \in [i-1]$, if $x_j \neq x_k$ then $y_j \neq y_k$, that is, the fixed oracle responses do not preclude an injective oracle.

Suppose that B_i never queries its oracle f on any $x \in \{0,1\}^\lambda$ s.t. $f(x) = y$ (in other words, $y \neq y_j$ for any $j \in [i-1]$). Then we argue that over the choice of $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, conditioned on (1) $f(x_j) = y_j$ for $j \in [i-1]$ and (2) y being in the image of f and (3) f being injective, it holds that $f(x') = y$ with probability at most $(i-1)/2^\lambda$. This corresponds exactly to the event $\Pr[\mathsf{Queried}_i \mid \mathsf{UnQueried}_{i-1}]$ in the statement of the claim. The inequality holds because at most $i-1$ rows of the truth table of f (on inputs of length λ) are fully fixed; thus, the preimage of y is distributed uniformly over the set $\{0,1\}^\lambda \setminus \{x_1, \ldots, x_{i-1}\}$ and equal to $x' \notin \{x_1, \ldots, x_{i-1}\}$ with probability $\leq 1/(2^\lambda - (i-1))$ when $i - 1 < 2^\lambda$. Note that since f is injective, y has at exactly one preimage. Further taking $i < 2^\lambda/2$ as specified in the statement of the subclaim, then $1/(2^\lambda - (i-1)) \leq 1/(2^\lambda - i) \leq 2/2^\lambda$, as desired. □

□

Given these claims, we are now ready to proceed to the proof of Theorem 4.4.

Proof of Theorem 4.4. Let n be any polynomial specified in the statement of the Theorem. Suppose that there is an ϵ-universal black-box reduction from $C^{(n)}$ to $C^{(1)}$ with oracle access to the primitive $\mathcal{F}$ comprising all functions $f : \{0,1\}^* \to \{0,1\}^*$. Namely, there is a PPT oracle machine R, such that for all $a \in [0,1]$, all functions $f : \{0,1\}^* \to \{0,1\}^*$, all augmented adversaries (A, Nat), and all $\lambda \in \mathbb{N}$, if (A, Nat) has robust advantage a for $C^{(n),f}$ on λ, then $(R^{A,f}, \mathsf{Nat})$ has robust advantage $\epsilon(\lambda, a)$ for $C^{(1),f}$ on λ.

We will show that there exists a negligible function μ such that $\epsilon(\lambda, a) \leq a + \mu(\lambda)$ for all security parameters $\lambda \in \mathbb{N}$ and all $a \in [0,1]$. The actual negligible function will be specified later on, but for now, it will suffice to require that $\mu(\lambda) \geq n \cdot \lambda^{\log \lambda}/2^\lambda$.

Consider some fixed advantage $a \in [0,1]$ and some security parameter λ. Whenever a security parameter λ has been fixed, we abuse notation and let n denote $n(\lambda)$. First note that if $a \geq 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$, then $a + \mu(\lambda) \geq 1$, so we trivially have that the reduction success's probability, $\epsilon(\lambda, a)$, is upper bounded by $a + \mu(\lambda)$. Thus, in the sequel we may assume without loss of generality that $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$. Fixing an $a \in [0,1]$ this means we only need to consider security parameters λ large enough to satisfy the inequality.

For every function f, and advantage a, we now construct an augmented adversary $(A, \mathsf{Nat}_{a,f})$. The rest of the proof proceeds as follows. Consider any $a \in [0,1]$ and $\lambda \in \mathbb{N}$ satisfying the condition that $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$. Then:

- We first show that for every *injective* function f, it holds that $(A, \mathsf{Nat}_{a,f})$ has robust advantage a for $C^{(n),f}$ on security parameter λ.
- By the correctness of the reduction R, it follows that for every injective f, $(R^{A,f}, \mathsf{Nat}_{a,f})$ must have robust advantage $\epsilon(\lambda, a)$ for $C^{(1),f}$ on λ.
- Since random length-tripling functions are injective with overwhelming probability (see Claim 4.5 above), it follows that there exists some fixed negligible function $\mu_1(\cdot)$ such that $(R^{A,f}, \mathsf{Nat}_{a,f})$ also has robust advantage $\epsilon(\lambda, a) - \mu_1(\lambda)$ for $C^{(1),f}$ on λ whenever f is *selected at random* from the set of length-tripling functions.
- We finally show that when f is selected at random, we can "simulate" the augmented attacker $(R^{A,f}, \mathsf{Nat}_{a,f})$ while ensuring that the simulation only fails with probability $a + \mu_2(\lambda)$, where $\mu_2(\cdot)$ is a negligible function. This concludes that we can invert f for a randomly selected function f with probability $\epsilon(\lambda, a) - a - \mu_1(\lambda) - \mu_2(\lambda)$. Moreover, the "simulator" makes at most a polynomial number of oracle queries, where the bound depends only on the starting universal reduction R.
- But since inverting a random oracle can only be done with negligible probability by any oracle attacker that makes a limited number of oracle queries (see Claim 4.6 above), it follows that there exists some fixed negligible function $\mu(\cdot)$ such that $\epsilon(\lambda, a) \leq a + \mu(\lambda)$, as desired. Moreover, this negligible function $\mu(\cdot)$ depends only on the number of oracle queries, and not on a.

Defining the augmented adversary $(A, \mathsf{Nat}_{a,f})$. Let A be the dummy attacker that forwards messages between the security game and $\mathsf{Nat}_{a,f}$. $\mathsf{Nat}_{a,f}$ given input 1^λ starts by initializing a list Seen to be empty, representing the set of queries it has previously seen and "saved" in memory. Let $\tilde{a} = a + n \cdot \lambda^{\log \lambda}/2^\lambda$ (i.e., think of $\tilde{a}$ as roughly a with some negligible amount added to it), and set $c = \max(0, n \cdot (1 - \tilde{a}^{1/n}))$. Subsequently, on input a message of the form $(y_1, \ldots, y_n) \in (\{0,1\}^{3\lambda})^n$, $\mathsf{Nat}_{a,f}$ does the following.

1. For each $i \in [n]$, $\mathsf{Nat}_{a,f}$ computes a response r_i as follows:
   (a) If $y_i \neq f(x')$ for some $x' \in \{0,1\}^\lambda$, $\mathsf{Nat}_{a,f}$ sets the temporary response r_i to be $\perp$.
   (b) If $y_i \in \mathsf{Seen}$, $\mathsf{Nat}_{a,f}$ sets r_i to be $\perp$.
   (c) Else, $\mathsf{Nat}_{a,f}$ sets r_i to be $\perp$ with probability c/n (independently for each i, where c is defined above based on a, λ, and n).[10]
   (d) If r_i has not been set to $\perp$, $\mathsf{Nat}_{a,f}$ sets r_i to be any value in $f^{-1}(y_i)$ of length λ.
   (e) $\mathsf{Nat}_{a,f}$ appends y_i to Seen. If $|\mathsf{Seen}| > \lambda^{\log \lambda}$, $\mathsf{Nat}_{a,f}$ removes the first (i.e. oldest) element from Seen. In other words, $\mathsf{Nat}_{a,f}$ is 'forgetful' and remembers only the $\lambda^{\log \lambda}$ most recent queries that it received.
2. If there exists an $i \in [n]$ such that $r_i = \perp$, $\mathsf{Nat}_{a,f}$ responds overall with $\perp$.
3. Otherwise, $\mathsf{Nat}_{a,f}$ responds with $(r_1, \ldots, r_n)$.

We highlight that we will analyze $\mathsf{Nat}_{a,f}$ exclusively in the regime where $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$ (without loss of generality by the above argument) in order for the probability c/n to be positive.

[10] Formally, Nat cannot sample a bit with probability c/n if c/n is not sampleable, but we can just sample a 2^{-b} approximation $\tilde{p}$ of c/n instead, for a sufficiently large b. See Remark 4.1 for full details.

Showing that $(A, \mathsf{Nat}_{a,f})$ has robust advantage a.. Choose any $a \in [0,1]$. In Claim 4.8 below, we show that for all λ satisfying $a < 1 - n \cdot \lambda^{\log \lambda}/2^{\lambda}$, for any choice of injective function $f : \{0,1\}^* \to \{0,1\}^*$, $(A, \mathsf{Nat}_{a,f})$ has robust advantage a for $C^{(n),f}$ on λ.

Claim 4.8. *For any injective function $f : \{0,1\}^* \to \{0,1\}^*$, for all $\lambda \in \mathbb{N}$ satisfying $a < 1 - n \cdot \lambda^{\log \lambda}/2^{\lambda}$, the augmented adversary $(A, \mathsf{Nat}_{a,f})$ has robust advantage a for $C^{(n),f}$ on λ.*

Proof. Fix an injective function $f : \{0,1\}^* \to \{0,1\}^*$, any $a \in [0,1]$, and let $(A, \mathsf{Nat}_{a,f})$ be the augmented adversary constructed as described previously. Consider λ s.t. $a < 1 - n \cdot \lambda^{\log \lambda}/2^{\lambda}$.

Let $\rho \in \{0,1\}^*$ be any interaction prefix. Suppose that on input 1^{λ} the security game $C^{(n),f}$ sends a random challenge $(y_1, \ldots, y_n) \leftarrow (f(U_\lambda))^n$, which A forwards to $\mathsf{Nat}_{a,f}(\rho)$. We note that $\mathsf{Nat}_{a,f}$ either responds with $(r_1, \ldots, r_n)$ such that $f(r_i) = y_i$ for all $i \in [n]$, which causes $C^{(n),g}$ to accept, or $\mathsf{Nat}_{a,f}$ responds with $\perp$. Thus, it suffices to upper bound the probability that $\mathsf{Nat}_{a,f}$ outputs $\perp$; we denote this event by $[\perp \leftarrow \mathsf{Nat}_{a,f}]$ for simplicity.

$\mathsf{Nat}_{a,f}$ outputs $\perp$ if for some i either (1) y_i is not in the image of f, or (2) $y_i \in \mathsf{Seen}$, or (3) if $\mathsf{Nat}_{a,f}$ randomly sets r_i to be $\perp$ (which happens independently for each i with probability c/n). The first case does not happen for honestly generated challenges, so we separately bound the probability for the second and third cases.

We start with analyzing the probability that $y_i \in \mathsf{Seen}$ for some i. Let $\mathsf{SomeSeen}$ denote the event that there is some i such that $y_i \in \mathsf{Seen}$. In other words, y_i is one of the $\lambda^{\log \lambda}$ most recent values that $\mathsf{Nat}_{a,f}$ previously saw in its initial view ρ or that $\mathsf{Nat}_{a,f}$ already processed as part of the current query. Note that each y_i is a random value in the image of f, which is of size 2^{λ} since f is injective, and moreover chosen by the challenger independently of the contents of Seen. So the probability that $y_i \in \mathsf{Seen}$ for any fixed i is at most $\lambda^{\log \lambda}/2^{\lambda}$. By a union bound, this implies that the probability there exists a $y_i \in \mathsf{Seen}$ is at most $n \cdot \lambda^{\log \lambda}/2^{\lambda}$.

Now, let us analyze the third case. Let Pass be the event that, for every $i \in [n]$, $\mathsf{Nat}_{a,f}$ sets r_i to be a value in $f^{-1}(y_i)$ of length λ. In other words, $\mathsf{Nat}_{a,f}$ did not abort for any i (and set r_i to be $\perp$). It follows that

$$\begin{aligned}
\Pr[\mathsf{Pass} \mid \neg\mathsf{SomeSeen}] &= \left(1 - \frac{c}{n}\right)^n \\
&= \left(1 - \frac{n \cdot (1 - \tilde{a}^{1/n})}{n}\right)^n \\
&= \tilde{a}.
\end{aligned}$$

Putting the above together, the probability that $\mathsf{Nat}_{a,f}$ outputs $\perp$ is bounded by

$$\begin{aligned}
\Pr[\perp \leftarrow \mathsf{Nat}_{a,f}] &= \Pr[\mathsf{SomeSeen}] + \Pr[\neg\mathsf{SomeSeen}] \cdot \Pr[\neg\mathsf{Pass} \mid \neg\mathsf{SomeSeen}] \\
&\leq \frac{n \cdot \lambda^{\log \lambda}}{2^{\lambda}} + (1 - \tilde{a}) \\
&= 1 - \left(a + \frac{n \cdot \lambda^{\log \lambda}}{2^{\lambda}}\right) + \frac{n \cdot \lambda^{\log \lambda}}{2^{\lambda}} \\
&= 1 - a.
\end{aligned}$$

□

Showing that $(R^{A,f}, \mathsf{Nat}_{a,f})$ succeeds for random f.. We now use the security of the reduction R (together with the fact that random expanding functions are injective with high probability as observed in Claim 4.5) to conclude that $(R^{A,f}, \mathsf{Nat}_{a,f})$ must have advantage $\epsilon(\lambda, a) - 2^{-\lambda}$ for $C^{(1),f}$ over the choice of a random $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$:

Claim 4.9. *For all a, λ s.t. $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$,*

$$\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_{a,f}) \text{ wins for } C^{(1),f} \text{ on } \lambda\right] \geq \epsilon(\lambda, a) - 2^{-\lambda}.$$

Proof. Consider any a, λ s.t. $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$. Recall that by Claim 4.8, for any injective function f, the augmented adversary $(A, \mathsf{Nat}_{a,f})$ has robust advantage a for $C^{(n),f}$ on λ. By the security of the the universal reduction, R, we have that $(R^{A,f}, \mathsf{Nat}_{a,f})$ must have robust advantage $\epsilon(\lambda, a)$ for $C^{(1),f}$ on λ. Thus,

$$\begin{aligned}
&\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_{a,f}) \text{ wins for } C^{(1),f} \text{ on } \lambda\right] \\
&\geq \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : f \text{ is injective}\right] \cdot \Pr\left[(R^{A,f}, \mathsf{Nat}_{a,f}) \text{ wins for } C^{(1),f} \text{ on } \lambda \mid f \text{ is injective}\right] \\
&\geq \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : f \text{ is injective}\right] \cdot \epsilon(\lambda, a) \\
&\geq (1 - 2^{-\lambda}) \cdot \epsilon(\lambda, a) \qquad \text{(by Claim 4.5)} \\
&\geq \epsilon(\lambda, a) - 2^{-\lambda}.
\end{aligned}$$

□

Simulating $(R^{A,f}, \mathsf{Nat}_{a,f})$ for random f. We construct a non-uniform oracle machine S that, using oracle access to any f, simulates the behavior of the augmented adversary $(R^{A,f}, \mathsf{Nat}_{a,f})$ in the view of the security game $C^{(1),f}$, but without requiring access to $\mathsf{Nat}_{a,f}$. S^f is defined as follows:

On input 1^λ, S^f runs a copy of R^A on 1^λ, and forwards messages between the security game and R. S^f must simulate 1) R's oracle queries to f, and 2) R's queries to $\mathsf{Nat}_{a,f}$. To simulate R's oracle queries to f, S^f:

1. Initializes a set $\mathsf{fQueries}$ to be empty, which represents all of the (input, output) pairs corresponding to queries to f that R makes.
2. On a query x, S^f computes $y = f(x)$ using its own oracle access to f, and adds (x, y) to the set $\mathsf{fQueries}$, and returns y.

To simulate queries to $\mathsf{Nat}_{a,f}$, S first initializes a list Seen to be empty, representing the set of queries it has previously seen. S takes the value c/n as advice (where c is the same used by $\mathsf{Nat}_{a,f}$), specified up to arbitrary precision. Now, to simulate the response to a query $(y_1, \ldots, y_n) \in (\{0,1\}^*)^n$:

1. For each $i \in [n]$, S computes a value of r_i as follows:
   (a) If $y_i \in \mathsf{Seen}$, or if $|y_i| \neq 3\lambda$, S sets r_i to be $\perp$.
   (b) S sets r_i to be $\perp$ with probability c/n (where c is the same as used by $\mathsf{Nat}_{a,f}$ above).
   (c) If r_i has not been set to $\perp$ and there exists an x' such that (x', y_i) is in $\mathsf{fQueries}$, S sets r_i to be x'.
   (d) Otherwise, S sets r_i to be $\perp$.

(e) S appends y_i to Seen. If $|\mathsf{Seen}| > \lambda^{\log \lambda}$, S removes the first (i.e. oldest) element from Seen.

2. If there exists an $i \in [n]$ such that $r_i = \bot$, S responds with $\bot$.

3. Otherwise, S responds with $(r_1, \ldots, r_n)$.

Claim 4.10. *Let $\ell(\cdot)$ be the polynomial that upper bounds the runtime of R. There exists a negligible function $\mu_3(\cdot)$ (depending on ℓ) s.t. for all a, λ s.t. $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$,*

$$\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : S^f \text{ wins for } C^{(1),f} \text{ on } \lambda\right] \geq \epsilon(\lambda, a) - a - \mu_3(\lambda).$$

Proof. Consider a, λ s.t. $a < 1 - n \cdot \lambda^{\log \lambda}/2^\lambda$. We first bound the probability that an execution $\langle C^{(1),f} \leftrightarrow S^f \rangle(1^\lambda)$ diverges from $\langle C^{(1),f} \leftrightarrow R^{A,f} \leftrightarrow \mathsf{Nat}_{a,f} \rangle(1^\lambda)$ on random $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$. Denote $\mathsf{fQueries}$ to be the set of queries that R makes to its function oracle f in either experiment. Recall that R also generates queries of the form $(y_1, \ldots, y_n)$ for A. Let y be the challenge that R receives from $C^{(1),f}$. In each experiment, define Bad to be the event that (at least) one of the following occurs:

1. **Non-Injectivity:** f is not injective,

2. **A Lucky Range Guess:** R at some point queries a $y_i \neq y$ s.t. $y_i \notin \mathsf{Seen}$, $y_i \notin \mathsf{fQueries}$, and $y_i = f(x')$ for some $x' \in \{0,1\}^\lambda$.

3. **Failure to Abort on y:** Whenever R queries some $(y_1, \ldots, y_n)$ s.t. $y = y_i$ for some $i \in [n]$ and $y_j \notin \mathsf{Seen}$ for all $j \in [n]$, then r_j is *not* set to $\bot$ for *any* choice of $j \in [n]$ during the step where either $\mathsf{Nat}_{a,f}$ or S sets $r_j \leftarrow \bot$ with probability c/n (independently for each $j \in [n]$).

Intuitively, the above three conditions determine exactly when the two experiments can diverge and thus we claim that

$$\begin{aligned}&\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_{a,f}) \text{ wins for } C^{(1),f} \text{ on } \lambda \mid \neg\mathsf{Bad}\right] \\ &= \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : S^f \text{ wins for } C^{(1),f} \text{ on } \lambda \mid \neg\mathsf{Bad}\right]\end{aligned}$$

To formalize this, consider any query $(y_1, \ldots, y_n)$ made by R and fix some list Seen (which will be updated the same way by $\mathsf{Nat}_{a,f}$ and S), and assume that Bad does not happen.

1. If for some $i \in [n]$, either (1) $|y_i| \neq 3\lambda$ or (2) y_i is not in the image of f or (3) $y_i \in \mathsf{Seen}$, then both $\mathsf{Nat}_{a,f}$ and S return $\bot$.

2. Thus, we concern ourselves only with queries where for all $i \in [n]$, $y_i = f(x_i)$ for some $x_i \in \{0,1\}^\lambda$ and $y_i \notin \mathsf{Seen}$. There are two cases: either $y_i \in \mathsf{fQueries}$ for all i, or $y_i \notin \mathsf{fQueries}$ for some i.

   (a) If $y_i \in \mathsf{fQueries}$ for all i, since, Bad does not happen and in particular the "non-injectivity condition", we have that f must be injective. Thus whenever S or Nat inverts y_i, they must return the same preimage (since the preimage is unique); moreover, whenever Nat successfully inverts, S is also able to do so by looking up the image in $\mathsf{fQueries}$.

(b) Else, there is some i s.t. $y_i \notin \mathsf{fQueries}$. S will return $\perp$ with probability 1 as it does not know the preimage of y_i, whereas $\mathsf{Nat}_{a,f}$ may in fact invert y_i. There are two cases to consider here too: either $y_i \neq y$, or $y_i = y$. If $y_i \neq y$, then the experiments must proceed exactly the same given the fact that the "A Lucky Guess" condition was not triggered (since Bad does not happen) Otherwise, if $y_i = y$, then since "Failure to Abort on y" condition was not triggered (since Bad does not happen), we again have that $\mathsf{Nat}_{a,f}$ does not not invert y_i so also here the experiments proceed exactly the same.

Finally, we claim that $\Pr[\mathsf{Bad}]$ is identical in both experiments since, as argued above, conditioned on Bad not happening, the experiment proceed in exactly the same way.

Let $\ell(\cdot)$ be the polynomial that bounds the runtime of R in the statement of the claim, and let $\ell = \ell(\lambda)$. In Claim 4.11, we show that there is a negligible function $\mu_r(\cdot)$ depending only on $\ell(\cdot)$ (and $n(\cdot)$) such that $\Pr[\mathsf{Bad}] \leq a + \mu_4(\lambda)$, and thus

$$\begin{aligned}
&\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : S^f \text{ wins for } C^{(1),f} \text{ on } \lambda\right] \\
&= \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_{a,f}) \text{ wins for } C^{(1),f} \text{ on } \lambda\right] - \Pr[\mathsf{Bad}] \\
&\geq \epsilon(\lambda, a) - 2^{-\lambda} - \Pr[\mathsf{Bad}] \qquad \text{(by Claim 4.9)} \\
&\geq \epsilon(\lambda, a) - a - 2^{-\lambda} - \mu_4(\lambda).
\end{aligned}$$

Taking $\mu_3(\lambda) = \mu_4(\lambda) + 2^{-\lambda}$ concludes the proof of the claim, as this function is negligible in λ. $\square$

Claim 4.11. *There exists a negligible function $\mu_4(\cdot)$ depending only on $\ell(\cdot)$ and $n(\cdot)$ s.t.* $\Pr[\mathsf{Bad}] \leq a + \mu_r(\lambda)$.

Proof. We first analyze the probability of each individual event comprising Bad:

1. **Non-injectivity:** By Claim 4.5, $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$ is not injective with probability upper bounded by $2^{-\lambda/2}$.

2. **A Lucky Range Guess:** Consider the probability (over $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$ and the coins of the execution) that R queries $(y_1, \ldots, y_n)$ s.t. for some $i \in [n]$, $y_i \neq y$ but $y_i \notin \mathsf{fQueries}$ and $y_i = f(x)$ for some $x \in \{0,1\}^\lambda$. That is, R finds an image of f on some input of length λ without having previously invoked its f-oracle on some preimage. We claim that the probability this occurs is at most $\ell \cdot n \cdot 2^{-\lambda}$.

   Fix any value y_i that R queries s.t. $y_i \notin \mathsf{fQueries}$ and $y_i \neq y$. For any fixed $x \in \{0,1\}^\lambda$, over the randomness of $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, $f(x)$ is uniform in $\{0,1\}^{3\lambda}$, and moreover independent of $f(x')$ for any $x' \neq x$, and thus independent of the view of R. Thus, the probability y_i is in the image of f (which is size at most 2^λ) is at most $2^\lambda \cdot 2^{-3\lambda}$. Taking a union bound over the polynomial ℓ number of queries R might make, and n values for y_i ($i \in [n]$) per query, then the probability that any one of any query of the form $(y_1, \ldots, y_n)$ is in the image of f is at most $(\ell \cdot n \cdot 2^\lambda) \cdot 2^{-3\lambda} \leq \ell \cdot n \cdot 2^{-\lambda}$.

3. **Failure to Abort on y:** Next, we deal with the case where R possibly sends a query $y_i = y$ as part of $(y_1, \ldots, y_n)$. In this case, S will output $\perp$ with probability 1. On the other hand, $\mathsf{Nat}_{a,f}$ will invert y assuming none of the other queries are set to $\perp$ and if $y \notin \mathsf{Seen}$. Recall from Claim 4.8 that we defined Pass to be the event that $\mathsf{Nat}_{a,f}$ does not set r_i to be $\perp$ with probability c/n for any i for a particular query. The same event can be defined for S

in identical fashion. In the event Pass, $\mathsf{Nat}_{a,f}$ will invert y whereas S will not, causing a divergence. We thus need to bound the probability of this event, but notice that we already did so in the proof of Claim 4.8; namely we have that $\Pr[\mathsf{Pass} \mid \neg\mathsf{SomeSeen}] = \tilde{a}$.

Finally, we need to bound the number of queries that contain y such that $y \notin \mathsf{Seen}$ when processed. Recall that as soon as a query containing y is received by either $\mathsf{Nat}_{a,f}$ or S, y is immediately added to the set Seen, and removed $\geq \lfloor \lambda^{\log \lambda} \rfloor$ queries later. Thus, in the sequence of ℓ queries that R might make, at most $1 + \lfloor \ell / \lambda^{\log \lambda} \rfloor$ queries contain y but $y \notin \mathsf{Seen}$; thus the overall probability of the "Failure to Abort on y" event is $\leq (1 + \lfloor \ell / \lambda^{\log \lambda} \rfloor) \cdot \tilde{a}$ by union bound.

We conclude the proof by applying a union bound,

$$\begin{aligned}
\Pr[\mathsf{Bad}] &\leq 2^{-\lambda} + \ell \cdot n \cdot 2^{-\lambda} + (1 + \lfloor \ell / \lambda^{\log \lambda} \rfloor) \cdot \tilde{a} \\
&\leq 2^{-\lambda} + \ell \cdot n \cdot 2^{-\lambda} + \tilde{a} + \lfloor \ell / \lambda^{\log \lambda} \rfloor \\
&= a + 2^{-\lambda} + n \cdot (\lambda^{\log \lambda} + \ell) \cdot 2^{-\lambda} + \lfloor \ell / \lambda^{\log \lambda} \rfloor
\end{aligned}$$

since recall that $\tilde{a} = a + n \cdot 2^{-\lambda} \cdot \lambda^{\log \lambda} < 1$. Taking $\mu_4(\lambda) = 2^{-\lambda} + n \cdot (\lambda^{\log \lambda} + \ell) \cdot 2^{-\lambda} + \lfloor \ell / \lambda^{\log \lambda} \rfloor$ concludes the proof, as this quantity is negligible in λ. □

Concluding the final bound.. Let $\ell(\cdot)$ be a polynomial that bounds the runtime of R. By Claim 4.10, there exists a negligible function $\mu_3(\cdot)$ depending only on $\ell(\cdot)$ s.t. choosing any a, λ satisfying $a < 1 - n \cdot \lambda^{\log \lambda} / 2^{\lambda}$, then for $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, S^f inverts f with probability $\epsilon(\lambda, a) - a - \mu_3(\lambda)$. By Claim 4.6, observing that S^f only queries f when $R^{A,f}$ queries f, and thus the number of queries made by S^f is bounded by the polynomial $\ell(\lambda)$,

$$\epsilon(\lambda, a) - a - \mu_3(\lambda) \leq 3 \cdot \ell(\lambda) \cdot 2^{-\lambda}$$

and thus

$$\epsilon(\lambda, a) \leq a + \mu_3(\lambda) + 3 \cdot \ell(\lambda) \cdot 2^{-\lambda}$$

Finally, choosing $\mu(\lambda) = \mu_3(\lambda) + 3 \cdot \ell(\lambda) \cdot 2^{-\lambda}$ concludes the proof of the Theorem. □

Remark 4.1. As previously noted in Footnote 10, formally we must account for values of c/n that are not sampleable. Instead we sample a "rounded" approximation of c/n, denote $\tilde{p}$, and proceed using $\tilde{p}$ in lieu of c/n. Here, take $b = 2n$ and thus $\tilde{p}$ will be a 2^{-b} approximation of c/n, namely $\frac{c}{n} - 2^{-b} \leq \tilde{p} \leq \frac{c}{n} + 2^{-b}$. To implement this, Nat will simply receive the first b bits of the binary expansion of c/n as advice (recalling that in our model Nature can be non-uniform). We then sample a bit with probability $\tilde{p}$ by using b random coins (e.g. see [HP15]). Like Nat, S also does this sampling by receiving the first b bits of the binary expansion of c/n as advice (and is thus non-uniform), and then efficiently samples the bit in exactly the same way as Nat.

To counteract the advantage loss due to rounding, we choose $c = n(1 - \tilde{a}^{1/n} - 2^{-b})$ instead. Then, in the proof of Claim 4.8, $\Pr[\mathsf{Pass} \mid \neg\mathsf{SomeSeen}] = (1 - \tilde{p})^n \geq (1 - \frac{c}{n} - 2^{-b})^n = (\tilde{a}^{1/n})^n = \tilde{a}$, as required to give $\mathsf{Nat}_{a,f}$ robust advantage a on $C^{(n),f}$ for all λ. For this choice of c, we consider only those choices of a and λ s.t. $\tilde{a}^{1/n} + 2^{-b} < 1$, or in other words $a < (1 - 2^{-b})^n - n\lambda^{\log \lambda} / 2^{\lambda} = 1 - \mu'(\lambda)$ for some negligible function μ'. For $a \geq 1 - \mu'(\lambda)$, again the theorem statement holds trivially. Later, for the analysis of the simulator in Claim 4.11, we can lower bound $\Pr[\mathsf{Pass} \mid \neg\mathsf{SomeSeen}] = (1 - \tilde{p})^n \leq (1 - \frac{c}{n} + 2^{-b})^n = (\tilde{a}^{1/n} + 2 \cdot 2^{-b})^n = \tilde{a} + \mu''(\lambda)$ for some negligible μ'' as required for bounding the probability of a "failure to abort on y".

4.4 Hardness Amplification for Re-randomizable One-Way Functions

Theorem 4.4 relies on the fact that the reduction works for *every* function f. We now show that if we only require hardness amplification for so-called *re-randomizable* functions, then the direct product construction works—in fact, even if setting the number of repetitions to 1. This result is interesting in the sense that the reduction we present is *not* single-shot straight line; rather, we do invoke the adversary many time, but can still argue that the reduction is universal. We additionally show that a single-shot straightline black-box reduction cannot be used. Thus, taken together, these results show that single-shot striaghline reductions are a strict subset of universal reductions.

We proceed to defining the notion of a re-randomizable function.

Defining Re-randomizable Functions. A re-randomizable function is closely related to the notion of re-randomizable encryption that has been considered [PR07, Gro04, CKN03]. We require that the output of a function can be re-randomized in a way that statistically hides the original input of the function. However, given a pre-image for the randomized output as well as the randomness used for re-randomization, you can recover the original input.

Definition 4.4 (Re-randomizable One-Way Function)**.** Let $\alpha\colon \mathbb{N} \to [0,1]$. A function f is a re-randomizable α-one-way function if it is an α-one-way function and there exist PPT algorithms rand, $\mathsf{recover}$ such that:

1. For all $\lambda \in \mathbb{N}$, $x \in \{0,1\}^\lambda$, it holds that
$$\left\{r \leftarrow \{0,1\}^\lambda : \mathsf{rand}(f(x), r)\right\} \equiv \left\{x' \leftarrow \{0,1\}^\lambda : f(x')\right\}.$$
2. For all $\lambda \in \mathbb{N}$, $x, r \in \{0,1\}^\lambda$, if $z \in f^{-1}(\mathsf{rand}(f(x), r))$ and $x' = \mathsf{recover}(z, r)$, then $f(x') = f(x)$.

While the above definition is strong, we note that the discrete-log based one-way function is re-randomizable in the common reference string model. Namely, if G is a group of prime order p and g a random generator G, then the function $f_{g,G}(x) = g^x$ is a re-randomizable function from $\mathbb{Z}_p$ to $\mathbb{Z}_p$. We can define $\mathsf{rand}(g^x, r)$ as $(g^x) \cdot (g^r) = g^{x+r}$, which is distributed identically to $g^{x'}$ for a random $x' \leftarrow \mathbb{Z}_p$. The we can define $\mathsf{recover}(z, r) = z - r$, which clearly satisfies the required correctness condition for $\mathsf{recover}$.

Hardness Amplification for Re-Randomizable Functions. We next show that if a function f is re-randomizable, then there is a trivial reduction that amplifies the success probability of any augmented adversary (A, Nat) for inverting f. In other words, any re-randomizable α-one-way function f where $\alpha \leq 1 - 1/\mathrm{poly}(\lambda)$ is a strong one-way function. Note that we do not even need to use a direct product construction to prove this (i.e. a 1-fold direct product suffices).

Theorem 4.12. *Let f be a re-randomizable one-way function with security game C. For any polynomial m, there exists an ϵ-universal reduction from C to C where $\epsilon(\lambda, a) = 1 - (1-a)^{m(\lambda)}$.*

Proof. Let $(A_{\mathsf{dummy}}, \mathsf{Nat})$ be an augmented adversary with robust advantage $a(\lambda)$ for any $\lambda \in \mathbb{N}$. For any choice of polynomial $m(\cdot)$, we will show an augmented adversary $(R_{\mathsf{dummy}}, \mathsf{Nat})$ with advantage $1 - (1 - a(\lambda))^{m(\lambda)}$ for C on any λ. By the Win Once Lemma 3.2 and the Dummy Lemma 3.3, we immediately get a full universal reduction.

Construct R_{dummy} as follows. On security parameter 1^λ and input $y = f(x)$ from the challenger C, R_{dummy} does the following:

1. First, R_{dummy} computes $m(\lambda)$ rerandomizations of y. Formally, for $i \in [m(\lambda)]$, R_{dummy} samples a random $r_i \leftarrow \{0,1\}^\lambda$ and sets $y_i = \mathsf{rand}(f(x), r_i)$.

2. For each $i \in [m(\lambda)]$, R_{dummy} first sends a $\mathsf{startsession}$ message to Nat, and then sends $(\text{`}\mathsf{att}\text{'}, y_i)$ to $\mathsf{Nat}$, receiving in response some $z_i$. (This choice of behavior is based on the construction of $A_{\mathsf{dummy}}$, which prepends its interaction with $\mathsf{startsession}$ and tags every message with `$\mathsf{att}$'.)

3. Finally, for each $i \in [m(\lambda)]$, R_{dummy} computes $x_i = \mathsf{recover}(z_i, r_i)$ and checks if $f(x_i) = y$. If so, R_{dummy} sends x_i back to the challenger.

Fix any $\lambda \in \mathbb{N}$. Next, for each $i \in [m(\lambda)]$, denote Invert_i the event that $z_i \in f^{-1}(y_i)$. We want to show that

$$\Pr[\mathsf{Invert}_i \mid \neg\mathsf{Invert}_1 \cap \ldots \cap \neg\mathsf{Invert}_{i-1}] \geq a(\lambda) \tag{1}$$

Fix any transcript ρ for Nat of the form

$$(\mathsf{startsession}, (y_1, z_1)), \ldots, (\mathsf{startsession}, (y_{i-1}, z_{i-1}))$$

such that $\neg\mathsf{Invert}_1 \cap \ldots \cap \neg\mathsf{Invert}_{i-1}$. Recall that by the robust advantage of $(A_{\mathsf{dummy}}, \mathsf{Nat})$, for any such prefix ρ, if $\mathsf{Nat}(\rho)$ receives in sequence a new $\mathsf{startsession}$ message followed by a $(\text{`}\mathsf{att}\text{'}, f(U_\lambda))$ message, then $\mathsf{Nat}(\rho)$ must invert $f(U_\lambda)$ with probability $\geq a(\lambda)$ (over the randomness of U_λ and Nat). Observing that $y_i \equiv f(U_\lambda)$ by the correctness of $\mathsf{rand}(\cdot)$, then Invert_i must occur with probability $\geq a(\lambda)$ on any such ρ, showing Equation 1.

Finally, Equation 1 immediately implies that

$$\begin{aligned}\Pr\left[\mathsf{Invert}_1 \cup \ldots \cup \mathsf{Invert}_{m(\lambda)}\right] &= 1 - \Pr\left[\neg\mathsf{Invert}_1 \cap \ldots \cap \neg\mathsf{Invert}_{m(\lambda)}\right] \\ &\geq 1 - (1 - a(\lambda))^{m(\lambda)}\end{aligned}$$

By the correctness of $\mathsf{recover}$, if $z_i \in f^{-1}(y_i)$ for any i, letting $x_i = \mathsf{recover}(z_i, r_i)$, then $f(x_i) = f(x)$ as required (where x is the original challenge sent to R_{dummy}), concluding the proof. □

Impossibility of Single-Shot Hardness Amplification. We proceed to showing that single-shot straightline black-box reductions cannot be used to prove a hardness amplification theorem through direct products even for re-randomizable functions, thus showing a separation between universal and single-shot straightline reductions.

Lemma 4.13. *For any polynomial n, for any function $f : \{0,1\}^* \to \{0,1\}^*$, suppose that there exists an ϵ-single-shot straightline black-box reduction from $C^{(n),f}$ to $C^{(1),f}$. Then there exist a PPT algorithm A' such that for every every sampleable probability a, $\epsilon^*(\lambda) = \{\epsilon(\lambda, a) - a\}$, it holds that A' has advantage $\epsilon^*(\cdot)$ for $C^{(1),f}$.*

Proof. Fix n and ϵ and $f : \{0,1\}^* \to \{0,1\}^*$. Suppose (by the definition of single-shot straightline black-box reduction) that there is a PPT oracle machine R s.t. for every A with some advantage a for $C^{(n),f}$ on some λ, $R^{f,A}$ has advantage $\epsilon(\lambda, a)$ for $C^{(1),f}$ on λ, and moreover R never restarts or rewinds A.

Now, fix any sampleable probability $a \in [0,1]$. Consider the machine B that on input 1^λ flips a biased coin that is heads with probability exactly a; if heads, B inverts the challenge from $C^{(n),f}$; if tails, B fails and returns $\perp$. Thus B has advantage a for $C^{(n),f}$ on any $\lambda \in \mathbb{N}$. Consider the PPT attacker machine B^* that always replies $\perp$ to every query that it receives. For any execution that

runs B at most once, and in a straightline way, with probability $1 - a$ over the randomness of the execution, B^* (when swapped for B) exactly simulates B. Thus for all $\lambda \in \mathbb{N}$:

$$\begin{aligned}\Pr\left[\mathsf{out}_{C^{(1),f}}[\langle C^{(1),f} \leftrightarrow R^{B^*}\rangle(1^\lambda)] = 1\right] &\geq \Pr\left[\mathsf{out}_{C^{(1),f}}[\langle C^{(1),f} \leftrightarrow R^{B}\rangle(1^\lambda)] = 1\right] - a \\ &\geq \epsilon(\lambda, a) - a\end{aligned}$$

and moreover, R^{B^*} is PPT. □

4.5 Impossibility of Hardcore Predicates

Recall that a predicate $h\colon \{0,1\}^\lambda \to \{0,1\}$ is *hardcore* for a function g if, for a random input x, $h(x)$ cannot be be predicted better than essentially a random guess given only $g(x)$. Formally, we capture this by a security game that samples x from the domain of g, sends $g(x)$, receives a bit b, and outputs 1 iff $b = h(x)$.

Recall that the Goldirech-Levin Theorem [GL89] shows that any one-way function f, can be slightly modified into a "randomness-extended" one-way function $g(x, r) = (f(x), r)$ such that the inner-product function $h(x, r) = \langle x, r\rangle$ is a hard-core predicate for g; in this construction, we have $|x| = |r|$. In other words, (the randomneness-extended version) of any one-way function f has a hard-core predicate (namely, the inner-product function). We now show that this construction cannot be proven secure using a universal reduction that only access the underlying function f as a black-box. In fact, we show an even stronger result, showing that no predicte h (no just the inner-product function) can be shown to be a hard-core predicate for g using a universal reduction that only uses f as a black-box. (Additionally, our result makes no assumption on the length of r in the construction of g from f.)

In more detail, let ℓ be a polynomial and denote by $C^f_{h,\ell}$ the security game that on input 1^λ first samples $(x, r) \leftarrow \{0,1\}^\lambda \times \{0,1\}^{\ell(\lambda)}$, then uses its oracle access to f to compute $g(x, r) = (f(x), r)$, sending it to the attacker, and on receiving a reply b outputs 1 iff $b = h(x, r)$. Denote C^f the one-way function security game for f. Note that both $C^f_{h,\ell}$ and C^f are PPT oracle machines, using only oracle access to f, as desired. As we did previously in Section 4.3, and with the same motivation, for each $\lambda \in \mathbb{N}$ denote $\mathcal{F}^{3\lambda}_\lambda$ the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$ s.t. $|f(x)| = 3|x|$ and for all $x \notin \{0,1\}^\lambda$, $f(x) = x \circ x \circ x$ where $\circ$ denotes concatenation.

Theorem 4.14 (Impossibility of a Goldreich-Levin Theorem)**.** *Let $\mathcal{F}$ be the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$. Let $\ell(\cdot)$ be any polynomial. Let $h : \{0,1\}^\lambda \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}$ be any function (taking any input length $\lambda \in \mathbb{N}$). Suppose there exists an ϵ-universal (black-box) reduction from $C_{h,\ell}$ to C with oracle access to the primitive $\mathcal{F}$. Then, there exists a negligible function μ such that $\forall \lambda \in \mathbb{N}, a < 1 - \lambda^{\log \lambda}/2^\lambda$, $\epsilon(\lambda, a) \leq \mu(\lambda)$.*

The proof of Theorem 4.14 follows a similar high-level structure as the proof for Theorem 4.4.

Proof. Let $\ell(\cdot)$ and $h(\cdot)$ be functions as specified in the Theorem statement. Suppose there exists an ϵ-universal black-box reduction from $C_{h,\ell}$ to C with oracle access to f—namely, that there is a PPT oracle machine R s.t. for all functions $f : \{0,1\}^* \to \{0,1\}^*$, all augmented adversaries (A, Nat), and all $\lambda \in \mathbb{N}$, if (A, Nat) has some robust advantage a for $C^f_{h,\ell}$ on λ, then $(R^{A,f}, \mathsf{Nat})$ has robust advantage $\epsilon(\cdot, a)$ for C^f on λ.

The proof proceeds according to the following high-level outline, in nearly identical fashion as for the proof of Theorem 4.4. First, for any function $f : \{0,1\}^* \to \{0,1\}^*$ we construct an augmented adversary (A, Nat_f). Now, consider any choice of $\lambda \in \mathbb{N}, a \in [0, 1]$ s.t. $a < 1 - \lambda^{\log \lambda}/2^\lambda$. Then:

- We first show that for every injective function f, the augmented adversary (A, Nat_f) has robust advantage $1 - \lambda^{\log \lambda}/2^\lambda$ for $C^f_{h,\ell}$ on λ. Thus, (A, Nat_f) immediately has robust advantage a for $C^f_{h,\ell}$ on λ.
- By the correctness of the universal reduction R, it follows that for every injective function f, $(R^{A,f}, \mathsf{Nat}_f)$ has advantage $\epsilon(\lambda, a)$ on C^f on λ.
- Since random length-tripling functions are injective with overwhelming probability (again, see Claim 4.5 in the previous section), it follows that there exists some fixed negligible function $\mu_1(\cdot)$ such that $(R^{A,f}, \mathsf{Nat}_f)$ also has robust advantage $\epsilon(\lambda, a) - \mu_1(\lambda)$ for C^f on λ whenever f is *selected at random* from the set of length-tripling functions.
- We finally show that when f is selected at random, we can "simulate" the augmented adversary $(R^{A,f}, \mathsf{Nat}_f)$ in polynomial time, while ensuring that the simulation aborts only $\mu_2(\lambda)$ of the time, where $\mu_2(\cdot)$ is a negligible function, and otherwise fails with probability $1/2$ independent of the success probability of the simulated reduction. This concludes that we can invert f in polynomial time for a randomly selected function f with probability $\frac{1}{2} \cdot \epsilon(\lambda, a) - \mu_2(\lambda)$.
- But since inverting a random oracle can only be done with negligible probability (see Claim 4.6 in the previous section), it follows that there exists some negligible function μ such that $\epsilon(\lambda, a) \leq \mu(\lambda)$, as desired.

Constructing the augmented adversary (A, Nat_f).. Let $f : \{0,1\}^* \to \{0,1\}^*$. A is the dummy adversary, and simply forwards messages between the security game and Nat_f. Nat_f on input 1^λ initializes a list Seen to be empty, representing the set of queries it has previously seen. Recall that the challenger $C^f_{h,\ell}$ will send the adversary queries of the form $(f(x), r)$ for $x \leftarrow \{0,1\}^\lambda, r \leftarrow \{0,1\}^{\ell(\lambda)}$, and that the adversary should respond with a guess for the hardcore bit $h(x, r)$. In our construction, on receiving any query $(y, r) \in \{0,1\}^{3\lambda} \times \{0,1\}^{\ell(\lambda)}$, Nat_f responds by doing the following:

1. If $y \notin \mathsf{Seen}$ and y is in the image of f, Nat_f finds some pre-image x s.t. $f(x) = y$ and sets $b = h(x, r)$.
2. Else, set b to be a random bit.
3. Finally, Nat_f responds with b and appends y to Seen. If $|\mathsf{Seen}| \geq \lambda^{\log \lambda}$, Nat_f removes the first (i.e. oldest) element from Seen.

Showing that (A, Nat_f) has robust advantage $1 - \lambda^{\log \lambda}/2^{\lambda+1}$ for $C^f_{h,\ell}$..

Claim 4.15. *For all injective functions f, for all $\lambda \in \mathbb{N}$, the augmented adversary (A, Nat_f) has robust advantage $1 - \lambda^{\log \lambda}/2^{\lambda+1}$ for $C^f_{h,\ell}$ on λ.*

Proof. Consider any injective function f, and any $\lambda \in \mathbb{N}$. Let $\rho \in \{0,1\}^*$ be any interaction prefix. Suppose that $C^f_{h,\ell}$ sends a random challenge $(y, r) \leftarrow (f(U_\lambda), U_{\ell(\lambda)})$, which A forwards to $\mathsf{Nat}_f(\rho)$. Note that y is always in the image of f. Now, there are two cases, corresponding to whether $y \in \mathsf{Seen}$. If $y \in \mathsf{Seen}$, Nat_f will output the wrong bit with probability $1/2$ over its random tape. If $y \notin \mathsf{Seen}$, then Nat_f will always output the correct bit.

We now analyze the probability than $y \in \mathsf{Seen}$. Since f is injective, its image on inputs of length λ must be size 2^λ, and moreover $y \leftarrow f(U_\lambda)$ must be uniformly distributed over these 2^λ possible values, and also independent of $y' \in \mathsf{Seen}$ (since it was generated by the challenger). Thus, the probability that $y \in \mathsf{Seen}$ (over the coins of the challenger) is at most $|\mathsf{Seen}|/2^\lambda \leq \lambda^{\log \lambda}/2^\lambda$.

Thus, the overall probability that $\mathsf{Nat}_f(\rho)$ outputs the wrong bit is

$$\begin{aligned} &= \Pr[y \in \mathsf{Seen}] \cdot 1/2 + \Pr[y \notin \mathsf{Seen}] \cdot 0 \\ &\leq \lambda^{\log \lambda}/2^{\lambda+1} \end{aligned}$$

Thus the probability that $\mathsf{Nat}_f(\rho)$ outputs the correct bit is at least $1 - \lambda^{\log \lambda}/2^{\lambda+1}$. □

Showing that $(R^{A,f}, \mathsf{Nat}_f)$ inverts random f.. We next show that the reduction $(R^{A,f}, \mathsf{Nat}_f)$ must invert a random $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$ with probability at least $\epsilon(\lambda, a) - 2^{-\lambda}$ for all a, λ s.t. $a < 1 - \lambda^{\log \lambda}/2^\lambda$.

Claim 4.16. *For all a, λ s.t. $a < 1 - \lambda^{\log \lambda}/2^\lambda$,*

$$\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_f) \text{ wins for } C^f \text{ on } \lambda\right] \geq \epsilon(\lambda, a) - 2^{-\lambda}$$

Proof. Consider any a, λ s.t. $a < 1 - \lambda^{\log \lambda}/2^\lambda$. Recall that by Claim 4.15), for any injective function f, the augmented adversary (A, Nat_f) has robust advantage $1 - \lambda^{\log \lambda}/2^\lambda$ for $C^f_{h,\ell}$ on λ, which immediately implies that it has robust advantage a for $C^f_{h,\ell}$ on λ, as $1 - \lambda^{\log \lambda}/2^\lambda > a$. By the security of the the universal reduction, R, we have that $(R^{A,f}, \mathsf{Nat}_f)$ must have advantage $\epsilon(\lambda, a)$ for C^f on λ, again for injective choices of f. Thus,

$$\begin{aligned} &\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : (R^{A,f}, \mathsf{Nat}_f) \text{ wins for } C^f \text{ on } \lambda\right] \\ &\geq \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : f \text{ is injective}\right] \cdot \Pr\left[(R^{A,f}, \mathsf{Nat}_f) \text{ wins for } C^f \text{ on } \lambda \mid f \text{ is injective}\right] \\ &\geq \Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : f \text{ is injective}\right] \cdot \epsilon(\lambda, a) \\ &\geq (1 - 2^{-\lambda}) \cdot \epsilon(\lambda, a) \qquad \text{(by Claim 4.5 from the previous section)} \\ &\geq \epsilon(\lambda, a) - 2^{-\lambda}. \end{aligned}$$

□

Simulating $(R^{A,f}, \mathsf{Nat}_f)$ for random f with failure probability $\leq 1/2$.. We construct a PPT oracle machine S that, using oracle access to f, attempts to simulates the behavior of the augmented adversary $(R^{A,f}, \mathsf{Nat}_f)$ in an interaction with the security game C^f, *without having access to* Nat_f. Subsequently, in Claim 4.17, we show that for random functions, S^f actually succeeds in performing this simulation with probability roughly $1/2$ and thus only has about a factor 2 loss in advantage for inverting f. Define S to be the following oracle machine:

On input 1^λ and oracle f, S^f runs a copy of $R^A(1^\lambda)$, and additionally simulates (1) R's oracle access to f as well as (2) responses for R's queries to Nature. To simulate R's oracle access to f:

1. S^f initializes a set $\mathsf{fQueries}$ to be empty, which represents all of the (input, output) pairs corresponding to queries to f that R makes.
2. On a query x, S^f computes $y = f(x)$ using its own oracle access to f, and adds (x, y) to the set $\mathsf{fQueries}$, and returns y.

To simulate responses to queries for Nat_f, S first initializes a list Seen to be empty, representing the set of queries it has previously seen. Now, to simulate the response to a query $(y, r) \in \{0,1\}^{3\lambda} \times \{0,1\}^{\ell(\lambda)}$:

1. If $y \notin \mathsf{Seen}$ and there exists x s.t. $(x, y) \in \mathsf{fQueries}$, S sets $b = h(x, r)$.
2. Else, set b to be a random bit.
3. Finally, S responds with b and appends y to Seen. If $|\mathsf{Seen}| \geq \lambda^{\log \lambda}$, S removes the first (i.e. oldest) element from Seen.

Note that S is PPT.

Claim 4.17. *Let $q(\cdot)$ be a polynomial that upper bounds the runtime of R. There exists a negligible function $\mu_2(\cdot)$ depending on $q(\cdot)$ s.t. for all a, λ where $a < 1 - \lambda^{\log \lambda}/2^\lambda$,*

$$\Pr\left[f \leftarrow \mathcal{F}_\lambda^{3\lambda} : S^f \text{ wins for } C^f \text{ on } \lambda\right] \geq \frac{1}{2} \cdot \epsilon(\lambda, a) - \mu_2(\lambda).$$

Proof. Fix any a, λ s.t. $a < 1 - \lambda^{\log \lambda}/2^\lambda$ as stipulated by the statement of the claim. Consider the two experiments

$$\left\{f \leftarrow \mathcal{F}_\lambda^{3\lambda} : \langle C^f \leftrightarrow S^f \rangle(1^\lambda)\right\}$$
$$\left\{f \leftarrow \mathcal{F}_\lambda^{3\lambda} : \langle C^f \leftrightarrow R^{A,f} \leftrightarrow \mathsf{Nat}_f \rangle(1^\lambda)\right\}$$

For any event X, denote $\Pr_S[X]$ and $\Pr_{\mathsf{Nat}}[X]$ to be the probability that X occurs in the corresponding experiment. At a high level, the two experiments may diverge in two different ways: first, an irrecoverable event Bad may occur in either experiment s.t. we are unable to bound its future behavior in any way. Second, R might ask a query (y, r) s.t. y is the challenge from C^f, in which case S will return the correct hardcore bit with independent probability $1/2$, whereas Nat_f will always return the correct answer; thus, conditioned on Bad not occurring, with probability $1/2$ the simulation is perfect.

We proceed to defining the event Bad. Denote $\mathsf{fQueries}$ to be the set of queries that R makes to its function oracle f in either experiment. Let y be the challenge output by the challenger C^f. Assume without loss of generality that R sends a query of the form (y, r) to Nature at some point during every execution. Define Bad to be the event that at least one of the following occurs, in the context of either experiment:

- **Non-injectivity.** f is not injective.
- **Lucky Range Guess.** R at some point queries (y', r) s.t. $y' \neq y$, $y' \notin \mathsf{fQueries}$, and $y' = f(x')$ for some $x' \in \{0,1\}^\lambda$.
- **Successful Repeated Queries.** R manages to sends *more than one* query of the form $(y, \cdot)$, (where, recall, y denotes the challenge received from C^f), such that $y \notin \mathsf{Seen}$ (and thus Nat_f would have provided an answer to the query).

We start by showing that this event happens with negligible probability.

Subclaim 4.18. *There is a negligible function $\mu_3(\cdot)$ (that depends only on R) s.t. for all λ,*

$$\Pr_{\mathsf{Nat}}[\mathsf{Bad}] \leq \mu_3(\lambda);$$
$$\Pr_{S}[\mathsf{Bad}] \leq \mu_3(\lambda).$$

Proof. Let $q(\cdot)$ be the polynomial that bounds the runtime of R, which is PPT; choose any $\lambda \in \mathbb{N}$ and denote $q = q(\lambda)$. We first analyze the probability of each individual event comprising Bad:

1. By Claim 4.5 (from the previous section), $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$ is not injective with $\Pr \leq 2^{-\lambda/2}$.
2. By the exact same argument as in Claim 4.11 from the previous section, taking $n = 1$, the probability of a "Lucky Range Guess" is $\leq q \cdot 2^{-\lambda}$.
3. Now, we analyze how often R sends more than one query of the form $(y, \cdot)$ such that $y \notin \mathsf{Seen}$. There are at most $1 + \left\lfloor q/\lambda^{\log \lambda} \right\rfloor$ such queries: as soon as a query containing y is received by either Nat_f or S, y is immediately added to the list Seen, and removed $\geq \lambda^{\log \lambda}$ queries later. Thus, $\left\lfloor q/\lambda^{\log \lambda} \right\rfloor$ (which is 0 whenever λ is sufficiently big) bounds the probability of a repeated successful query.

We conclude the proof by taking a union bound over all three events,

$$\Pr[\mathsf{Bad}] \leq 2^{-\lambda} + q \cdot 2^{-\lambda} + \left\lfloor q/\lambda^{\log \lambda} \right\rfloor. \qquad \square$$

We next analyze the probability that the simulation succeeds *conditioned* on Bad not occurring. Towards this, let $\mathsf{GoodGuess}$ to be the event that

- S^f **correctly responds to** (y, r)**.** When R sends *for the first time* a query of the form (y, r) for some $r \in \{0,1\}^{\ell(\lambda)}$ s.t there is no x where $(x, y) \in \mathsf{fQueries}$, R receives in response the correct $b = h(y, r)$.

Note that this event happens with probability $1/2$. It will be roughly $1/2$ even if we condition on Bad not occurring: Thus we directly have the following claim:

Subclaim 4.19. *Let μ_3 be the negligible function guaranteed to exists by Subclaim 4.18. Then, the following holds:*

$$\Pr_S[\mathsf{GoodGuess} \mid \neg\mathsf{Bad}] \geq \frac{1}{2} - \mu_3(\lambda)$$

Proof. As argued above, we have that $\Pr_S[\mathsf{GoodGuess}] = \frac{1}{2}$. Since by Subclaim 4.18, $\Pr[\mathsf{Bad}] \leq \mu_3(\lambda)$, it directly follows that

$$\Pr_S[\mathsf{GoodGuess} \mid \neg\mathsf{Bad}] \geq \Pr_S[\mathsf{GoodGuess}] - \Pr[\mathsf{Bad}] \geq \frac{1}{2} - \mu_3(\lambda)$$

$\square$

We now have the following claim which argues that if $\mathsf{GoodGuess}$ happens, then the simulation is done perfectly.

Subclaim 4.20. *The following holds:*

$$\Pr_S[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad} \cap \mathsf{GoodGuess}] = \Pr_{\mathsf{Nat}}[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}].$$

Proof. Consider either the experiment for S or the experiment for Nat. If we are analyzing the experiment for S, assume that the event $\mathsf{GoodGuess}$ occurs. Fix some list Seen (which will be updated the same way by Nat_f and S), and assume that the event Bad has not and will never occur. Now, suppose that R makes a query $(y', r) \in \{0,1\}^\lambda \times \{0,1\}^{\ell(\lambda)}$ – we now analyze the resulting behavior, and argue that it is the same for both experiments:

1. If for some $i \in [n]$, either (1) y' is not in the image of f or (2) $y' \in \mathsf{Seen}$, then both Nat_f and S return a random bit.

2. Thus, we concern ourselves only with queries where y' is in the image of f and $y' \notin \mathsf{Seen}$. There are two cases: either $y' \in \mathsf{fQueries}$ or $y' \notin \mathsf{fQueries}$.

   (a) If $y' \in \mathsf{fQueries}$, since Bad doesn't happen, we know that f is injective by virtue of the "non-injectivity condition". Thus, whenever S or Nat_f responds to (y', r), they must return the same hardcore predicate $h(f^{-1}(y'), r)$ (as the preimage $f^{-1}(y')$ is unique). Moreover, whenever Nat_f successfully computes the correct hardcore bit, S also does so by looking up the preimage of y' in $\mathsf{fQueries}$.

   (b) If $y' \notin \mathsf{fQueries}$, S will always return a random bit, whereas $\mathsf{Nat}_{a,f}$ may in fact return the correct hardcore bit. There are two cases here to consider: either $y' \neq y$, or $y' = y$. Since we know assumed that Bad doesn't happen, that immediately rules out $y' \neq y$ (as otherwise it triggers the Bad "lucky range guess" condition).

   Otherwise, if $y' = y$:

   - By virtue of the "Successful Repeated Queries" condition in Bad, and since Bad does not happen by assumption, we know that y cannot have been sent by R in a previous query.
   - Thus, y must have been sent by R for the first time. Nat_f will always return the correct bit. Because we assumed that the event $\mathsf{GoodGuess}$ occurs, S must also return the correct bit. □

Putting it all together:

$$
\begin{aligned}
&\Pr_S[C^f \text{ outputs "win"}] \\
&\geq \Pr_S[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}] - \Pr_S[\mathsf{Bad}] \\
&\geq \Pr_S[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}] - \mu_3(\lambda) && \text{(by Subclaim 4.18)} \\
&\geq \Pr_S[\mathsf{GoodGuess} \mid \neg\mathsf{Bad}] \cdot \Pr_S[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}, \mathsf{GoodGuess}] - \mu_3(\lambda) \\
&= \left(\frac{1}{2} - \mu_3(\lambda)\right) \cdot \Pr_S[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}, \mathsf{GoodGuess}] - \mu_3(\lambda) && \text{(by Subclaim 4.19)} \\
&\geq \left(\frac{1}{2} - \mu_3(\lambda)\right) \cdot \Pr_{\mathsf{Nat}}[C^f \text{ outputs "win"} \mid \neg\mathsf{Bad}] - \mu_3(\lambda) && \text{(by Subclaim 4.20)} \\
&\geq \frac{1}{2} \Pr_{\mathsf{Nat}}[C^f \text{ outputs "win"}] - \Pr_{\mathsf{Nat}}[\mathsf{Bad}] - 2\mu_3(\lambda) \\
&\geq \frac{1}{2} \Pr_{\mathsf{Nat}}[C^f \text{ outputs "win"}] - 3\mu_3(\lambda) && \text{(by Subclaim 4.18)} \\
&\geq \frac{1}{2}\epsilon(\lambda, a) - 2^{-\lambda} - 3\mu_3(\lambda) && \text{(by Claim 4.16)} \\
&= \frac{1}{2} \cdot \epsilon(\lambda, a) - \mu_2(\lambda)
\end{aligned}
$$

where $\mu_2(\lambda) = 3\mu_3(\lambda) + 2^{-\lambda-1}$, which is a negligible function. □

Concluding the final bound.. By Claim 4.17, there exists a negligible function $\mu_2(\cdot)$ s.t. choosing any a, λ satisfying $a < 1 - \lambda^{\log \lambda}/2^\lambda$, then for $f \leftarrow \mathcal{F}_\lambda^{3\lambda}$, S^f inverts f with probability $\frac{1}{2} \cdot \epsilon(\lambda, a) - \mu_2(\lambda)$. By Claim 4.6 from the previous section, random functions f cannot be inverted with non-negligible probability:

$$\frac{1}{2} \cdot \epsilon(\lambda, a) - \mu_2(\lambda) \leq 3 \cdot \ell \cdot 2^{-\lambda}$$

and thus

$$\epsilon(\lambda, a) \leq 2 \cdot (\mu_2(\lambda) + 3 \cdot \ell \cdot 2^{-\lambda}).$$

Let $\mu'(\lambda) = 2 \cdot (\mu_2(\lambda) + 3 \cdot \ell \cdot 2^{-\lambda})$, which is a negligible function. As argued at the beginning of the proof, taking $\mu(\lambda) = \mu'(\lambda)$ for all choices of λ where $\lambda^{\log \lambda}/2^\lambda \leq 0.01$, and $\mu(\lambda) = 1$ otherwise, then for all $a \in [0, 0.99]$ and $\lambda \in \mathbb{N}$,

$$\epsilon(\lambda, a) \leq \mu(\lambda)$$

which finally shows the Theorem. □

5 Subclasses of Augmented Adversaries

The definitions of universal reductions we have seen so far are very strong. For the basic definition, we make no assumptions on the behavior of the augmented adversary (A, Nat), other than the fact that (A, Nat) will win security games with some advantage over honestly generated challenges, for any interaction prefix for Nat.

We now consider restrictions on augmented adversaries, limiting the power of Nature, and show that once we do so, we can overcome the impossibility results presented in Sections 4.3 and 4.5. In particular, we consider "forgetful" Natures that can evolve over time, but only "remember" the contents of the last k messages that it has seen. Looking ahead, we note that for the main definition that we consider—*universal reductions w.r.t. time-evolving k-window Natures*—we will not show that the classic reductions for proving hardness amplification [Yao82] or the security of the Goldreich-Levin hard-core bit [GL89] work, but rather demonstrate *new* reductions (unfortunately, with higher running time). In fact, we will demonstrate that for a large class of so-called non-adaptive straight-line black-box reduction (into which, e.g., the reductions of [Yao82] and [GL89] fall into), we can *transform* such reductions into universal reductions for this relaxed setting.

The notion of a universal reductions w.r.t. time-evolving k-window Natures will be defined exactly like a universal reduction, except that we will restrict our attention only to the subclass of *time-evolving k-window* Natures for each k. Towards defining this notion, as a warm-up, we will first consider an even more (and perhaps overly) restrictive class of *k-window* adversaries.

5.1 Warm Up: k-Window Natures

In this section, as a warm-up, we consider *k-window Natures*. Roughly speaking, a *k-window* Nature Nat remembers only the last k queries that it has received. In essence, such a Nat keeps enough state to play up to a k-round security game. Later we shall see that such a Nature can otherwise be treated as a stateless algorithm that can be "restarted" by sending it, for instance, k prespecified "dummy" messages in advance; indeed, this property is what makes the definition attractive.

Recall that for an interaction prefix $\rho \in \{0,1\}^*$, we use $\|\rho\|$ to denote the number of queries contained within ρ:

Definition 5.1 (k-Window Natures)**.** Let $\mu : \mathbb{N} \to [0,1]$ and let $k : \mathbb{N} \to \mathbb{N}$ be a polynomial function. We say that Nat is a (k,μ)-*window Nature* if for all $\lambda \in \mathbb{N}$, for all $\rho, \rho', \rho'' \in \{0,1\}^*$ where $\|\rho''\| = k(\lambda)$, and for all interactive machines O,

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho \circ \rho'')\rangle(1^\lambda)], \\ \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho' \circ \rho'')\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda).$$

Finally, we say that Nat is simply a k-*window* Nature if there exists a negligible function μ s.t. Nat is a (k,μ)-window Nature.

To better understand this definition, note that for any two query prefixes, as long as they share the same last $k(\lambda)$ queries (and randomness used to respond to those queries), then a k-*window* Nature must behave the same way on those two prefixes, even if they are different lengths. This captures the idea that Nature clearly can keep state within a single session of a security game (i.e., in an up-to-$k(\lambda)$-round interaction with A) up to a certain point (i.e., if A sends at most $k(\lambda)$ queries), but becomes stateless across multiple sessions with security games.

Resetting Nature. A corollary of our k-window Nature definition is that by sending Nature $k(\lambda)$ "dummy messages", say $\perp$ messages, it is possible to "reset" the state of Nature to what is essentially a state independent of prior interactions. This follows because a k-window Nature must forget interactions seen more than $k(\lambda)$-queries ago when run on some security parameter 1^λ.

Lemma 5.1 (Resetting Nature By Sending k Dummy Messages)**.** *Let $\mu : \mathbb{N} \to [0,1]$, let $k : \mathbb{N} \to \mathbb{N}$ be a polynomial function, and let (A, Nat) be an augmented adversary s.t. Nat is a (k,μ)-window Nature. Define A_{reset} to be the attacker that on input 1^λ, first sends Nature $k(\lambda)$ number of dummy messages $\perp$, and then subsequently runs $A(1^\lambda)$ to interact with both the challenger and Nature. Then, for any challenger C, for all $\lambda \in \mathbb{N}$, for all prefixes $\rho, \rho' \in \{0,1\}^*$,*

$$\bigtriangleup \begin{pmatrix} \mathit{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)], \\ \mathit{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho')\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda).$$

Proof. For any random tape $r \in \{0,1\}^*$, and for any prefix $\rho'' \in \{0,1\}^*$ denote $\mathsf{Nat}(\rho''; r)$ the machine with its read-once random tape fixed to r after processing ρ'' (that is, the next read location is the first bit of r). Now, let ρ, ρ' be two prefixes that do not necessarily need to be the same length, as specified in the lemma statement. Choose any $\lambda \in \mathbb{N}$, and any $r \in \{0,1\}^*$. Denote $\rho^r = (r, \perp, \perp, \ldots, \perp)$ s.t. $\|\rho^r\| = k(\lambda)$. Then,

$$\begin{aligned}
&\Pr[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho; r)\rangle(1^\lambda)] = 1] && (2)\\
&\quad = \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho \circ \rho^r)\rangle(1^\lambda)] = 1\big] && (3)\\
&\quad \geq \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho' \circ \rho^r)\rangle(1^\lambda)] = 1\big] - \mu(\lambda) && (4)\\
&\quad = \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho'; r)\rangle(1^\lambda)] = 1\big] - \mu(\lambda) && (5)
\end{aligned}$$

where the inequality follows directly from the (k,μ)-window property of Nat.

Now, let b be an upper bound on the number of bits of additional randomness that Nat will use in either the interaction $\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)$ or $\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho')\rangle(1^\lambda)$, that is not previously encoded by ρ or ρ'. This bound exists because we model Nat as an ITM that halts on its input. Thus, each possible tape $r \in \{0,1\}^b$ is equally likely to be used by Nat to finish the interaction. Now we can write:

$$
\begin{aligned}
&\Pr[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)] = 1] - \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho')\rangle(1^\lambda)] = 1\big] \\
&\quad = \sum_{r\in\{0,1\}^b} 2^{-b} \cdot \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho; r)\rangle(1^\lambda)] = 1\big] \\
&\qquad\quad - \sum_{r\in\{0,1\}^b} 2^{-b} \cdot \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho'; r)\rangle(1^\lambda)] = 1\big] \\
&\quad = \sum_{r\in\{0,1\}^b} 2^{-b} \cdot (\Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho; r)\rangle(1^\lambda)] = 1\big] \\
&\qquad\qquad\qquad - \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho'; r)\rangle(1^\lambda)] = 1\big]) \\
&\quad \leq \sum_{r\in\{0,1\}^b} 2^{-b} \cdot \mu(\lambda) \\
&\quad = \mu(\lambda).
\end{aligned}
$$

□

Universal Reductions for k-Window Natures. We now turn to defining universal reductions with respect to k-window Natures, simply by restricting attention to such Natures.

Definition 5.2 (Universal Reductions for k-Window Natures)**.** Let $k \in \mathbb{N}$, $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$, $\mu : \mathbb{N} \to [0,1]$, and let C, C' be security games. We say that there is a *ϵ-universal reduction* from C to C' *w.r.t.* *(k,μ)-window Natures* if for all PPT A, there exists PPT A' s.t. for every (k,μ)-window Nature Nat, letting $a(\cdot)$ denote (A, Nat)'s robust advantage for C, then (A', Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'.

On Dummy Adversaries. Observe that the Dummy Lemma (see Theorem 3.3) no longer holds directly when we restrict to k-window Natures. The reason is that, fixing some polynomial $k : \mathbb{N} \to \mathbb{N}$, although a k-window Nature Nat may forget messages that are too old, an accompanying attacker $A(1^\lambda)$ could in fact keep state for more than $k(\lambda)$ rounds. Thus, a combined Nature that combines both A and Nat internally must emulate the statefulness of A, and may no longer be a k-window Nature. Thus we cannot immediately apply a universal reduction for k-window Natures (and the dummy attacker) to this combined Nature.

Universal Reductions for k-Window Natures from Classical Reductions. Since we are imposing quite strong restrictions on Nature (but note that these restrictions are still a lot less restrictive than those classically used in the cryptographic literature) it should not be a surprise that classic reductions, at least of the straight-line black-box type, directly yield universal reductions with respect to this class of attacker. To formalize this, we need to restrict our attention to straight-line black-box reductions that only *sequentially* invoke the attacker:

We start by defining such *sequential*, straightline, black-box reductions.

Definition 5.3 (Sequential Straightline Black-box Reductions)**.** Let C, C' be security games, $a\colon \mathbb{N} \to [0,1]$, $\epsilon\colon \mathbb{N} \times [0,1] \to [0,1]$. We say that there is an ϵ-sequential straightline black-box reduction from C to C' if there exists a uniform PPT oracle machine R such that for every adversary A with advantage $a(\lambda)$ for C on λ, $R^A(1^\lambda)$ has advantage $\epsilon(\lambda, a(\lambda))$ for C' on λ after with A. Furthermore, R may never rewind its oracle, but may restart it oracle an arbitrary number of time (but if it does so, it can not go back to a previous execution).

We next observe that such a reduction directly yields the existence of a universal reduction w.r.t. k-window Natures. At a high level, the argument is as follows. Given a sequential straight-line black box reduction R, construct a universal reduction that is essentially the same, except each time R invokes its oracle, the universal reduction interacts with Nat instead, using an internal copy of the attacker A_{reset} (mimicking $R \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}$). Each time R *restarts* its oracle, the universal reduction restarts its copy of A_{reset} instead, thus sending Nat $k(\lambda)$ "dummy queries" to force Nat to forget prior state (on a security parameter λ). Because Nat is a k-window Nature, we can show that each interaction with a copy of A_{reset} can be simulated by some fixed algorithm S (without rewinding Nat). Thus, the universal reduction behaves like R^S, by a hybrid argument, where we substitute S for each session with $A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}$, one at a time.

Lemma 5.2. *Let $\epsilon : \mathbb{N} \times [0,1] \to [0,1]$, $q : \mathbb{N} \to \mathbb{N}$, and C, C' be security games. Suppose there exists a ϵ-sequential straight-line black-box reduction from C to C', that on input 1^λ restarts its oracle at most $q(\lambda)$ times. Then for every function $\mu : \mathbb{N} \to [0,1]$, for every polynomial $k : \mathbb{N} \to \mathbb{N}$, there is a ϵ^*-universal reduction from C to C' w.r.t. (k,μ)-window Natures, where $\epsilon^*(\lambda, a) = \epsilon(\lambda, a) - q \cdot \mu$.*

We include the detailed proof (as well as other basic observations about k-window Natures, such as composition) in Appendix A, as they use standard techniques, or are similar to previous proofs.

5.2 Time-Evolving k-Window Natures

We now consider a more relaxed class of Natures that we refer to as *time-evolving k-window Natures*. Such Natures remember only the last $k(\lambda)$ messages that it received (when run on some security parameter 1^λ), but may also "age" and depend on the number of queries that they have received (but not the contents of queries that are more than $k(\lambda)$ messages old). This is in contrast to simply *k-window Natures* (Section 5.1) that do not remember how many queries that they have received, that is, 'keep track of time'.

Definition 5.4 (Time-Evolving k-Window Natures)**.** Let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$, and let $\mu : \mathbb{N} \to [0,1]$. We say that Nat is a *time-evolving (k,μ)-window Nature* if for all $\lambda \in \mathbb{N}$, for all $\rho, \rho', \rho'' \in \{0,1\}^*$ where $\|\rho\| = \|\rho'\|$ and $\|\rho''\| = k(\lambda)$, and for all interactive machines O,

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho \circ \rho'')\rangle(1^\lambda)], \\ \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho' \circ \rho'')\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda).$$

We say that Nat is simply a *time-evolving k-window* Nature if there exists a negligible function μ s.t. Nat is a time-evolving (k,μ)-window Nature.

The definition above is exactly the same as the definition for k-window Natures, Definition 5.1, except that we quantify over only those prefixes ρ and ρ' of the same length. Thus, in contrast to the k-window setting, in the time-evolving setting Nat can change its behavior as it receives a larger

number of queries—but not based on the contents of those queries, except for the last k queries. This models the fact that an augmented adversary may have different behavior over time (where we formalize time as the number of queries that Nat has received) as it plays multiple security games, but where its evolution is independent of the actual outcomes of prior security games.

Again, for any polynomial function $k(\cdot)$, we define universal reductions w.r.t. time-evolving k-window Natures in the same way as universal reductions, except that we restrict our attention to time-evolving k-window Natures.

Definition 5.5 (Universal Reductions for Time-Evolving k-Window Natures)**.** Let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$, $\mu : \mathbb{N} \to [0,1]$, $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, and let C, C' be security games. We say that there is a *ϵ-universal reduction* from C to C' *with respect to time-evolving (k,μ)-window Natures* if for all PPT A, there exists PPT A' such that for every time-evolving (k,μ)-window Nature Nat, letting $a(\cdot)$ denote (A, Nat)'s robust advantage for C, then (A', Nat) has robust advantage $\epsilon(\cdot, a(\cdot))$ for C'.

Win-Once Universal Reductions. When restricting to time-evolving k-window Natures, "win once" universal reductions continue to imply universal reductions.

Definition 5.6 (Win-once Universal Reductions for Time-Evolving k-Window Natures)**.** Let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$, $\mu : \mathbb{N} \to [0,1]$, $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, C and C' be security games. We say that there is a *win-once ϵ-universal reduction* from C to C' *w.r.t. time-evolving (k,μ)-window Natures* if for all PPT A, there exists a PPT A' such that for every time-evolving (k,μ)-window Nature Nat, letting $a(\cdot)$ denote (A, Nat)'s robust advantage for C, then (A', Nat) has simply advantage $\epsilon(\cdot, a(\cdot))$ for C'.

Lemma 5.3 (Win-once Universal Reductions imply Universal Reductions for Time-Evolving k-Window Natures)**.** *Let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$, $\mu : \mathbb{N} \to [0,1]$, $\epsilon \colon \mathbb{N} \times [0,1] \to [0,1]$, and C and C' be security games. If there exists a* win-once ϵ-universal reduction *from C to C'* w.r.t. time-evolving (k,μ)-window Natures*, then there exists a* ϵ-universal reduction *from C to C'* w.r.t. time-evolving (k,μ)-window Natures*.*

The proof is nearly identical to that for the generic setting, with the addition that we must argue that the Nature that hardcodes receiving a prefix ahead of time is itself a time-evolving k-window Nature:

Proof. Assume that there exists a win-once ϵ-universal reduction from C to C' w.r.t. time-evolving (k,μ)-window Natures. Thus, for any A there exists A' s.t. for any time-evolving (k,μ)-window Nature Nat, letting $a(\cdot)$ be (A, Nat)'s robust advantage for C, then (A', Nat) has advantage $\epsilon(\cdot, a(\cdot))$ for C'. We claim that (A', Nat) also has robust advantage $\epsilon(\cdot, a(\cdot))$ for C': For any interaction prefix $\rho \in \{0,1\}^*$, denote Nat_ρ the Nature machine that on input 1^λ, simply runs Nat on input 1^λ, but starting in the state where Nat already saw the prefix ρ. We stress that Nat_ρ hardcodes ρ (as a non-uniform advice string). Then, for any prefix ρ, (A, Nat_ρ) itself also has robust advantage $a(\cdot)$ for C; namely, for all $\rho' \in \{0,1\}^*$,

$$\begin{aligned}\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}_\rho(\rho')\rangle(1^\lambda)] = 1\right] &= \Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho \circ \rho')\rangle(1^\lambda)] = 1\right] \\ &\geq a(\lambda).\end{aligned}$$

The inequality follows from the robust advantage of (A, Nat) for C, as $\rho \circ \rho'$ itself is a valid prefix.

Importantly, for any prefix ρ, Nat_ρ is a time-evolving (k,μ)-window Nature, which follows from the fact that Nat is a time-evolving (k,μ)-window Nature. To see why, observe that the latter immediately implies that for all $\rho \in \mathbb{N}$, for all $\lambda \in \mathbb{N}$, for all $\rho', \rho'', \rho^* \in \{0,1\}^*$ where $\|\rho'\| = \|\rho''\|$ and $\|\rho^*\| = k(\lambda)$, $\forall O$,

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho \circ \rho' \circ \rho^*)\rangle(1^\lambda)], \\ \mathsf{out}_O[\langle O \leftrightarrow \mathsf{Nat}(\rho \circ \rho'' \circ \rho^*)\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda), \tag{6}$$

because $\|\rho \circ \rho'\| = \|\rho \circ \rho''\|$ and both are valid prefixes for Nat. Equation 6 coincides exactly the definition of the time-evolving (k,μ)-window property for Nat_ρ.

Thus we can apply the win-once ϵ-universal reduction w.r.t. time-evolving (k,μ)-window Natures to (A, Nat_ρ). Consequently, for any prefix ρ, (A', Nat_ρ) has advantage $\epsilon(\cdot, a(\cdot))$ for C'. Thus, for any prefix ρ, and any security parameter λ:

$$\begin{aligned} \Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)] = 1\right] &= \Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}_\rho\rangle(1^\lambda)] = 1\right] \\ &\geq \epsilon(\lambda, a(\lambda)) \end{aligned}$$

concluding the proof. □

Resetting Nature. As is the case in the k-window setting (that is, not time-evolving; see Lemma 5.1), we note that by sending Nature $k(\lambda)$ "dummy messages", it is possible to "reset" the state of Nature to what is essentially a state independent of prior interactions (but now Nature can still evolve based on time, or the number of queries that it has received). This follows because a (time-evolving) k-window Nature must forget interactions seen more than $k(\lambda)$-queries ago.

Lemma 5.4. *Let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$, let $\mu : \mathbb{N} \to [0,1]$, and let (A, Nat) be an augmented adversary s.t. Nat is a time-evolving (k,μ)-window Nature. Define A_{reset} to be the attacker that on input 1^λ, first sends Nature $k(\lambda)$ number of dummy messages $\perp$, and then subsequently runs $A(1^\lambda)$ to interact with both the challenger and Nature. Then, for any challenger C, for all $\lambda \in \mathbb{N}$, for all prefixes $\rho, \rho' \in \{0,1\}^*$ s.t. $\|\rho\| = \|\rho'\|$,*

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)], \\ \mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho')\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda).$$

Proof. The proof follows identically from that of Lemma 5.1, the only difference being that the two prefixes ρ and ρ' must now be the same length. Equation 4 (in the proof of Lemma 5.1) then follows from the 'less constraining' time-evolving (k,μ)-window property of Nat, instead of requiring the stronger (k,μ)-window property. This is because we no longer need to deal with arbitrary length prefixes ρ, ρ'; the two prefixes are now of the same length. The remainder of the proof of Lemma 5.1 can be applied to the current setting verbatim. □

Composability. Composability for universal reductions w.r.t. time-evolving (k,μ)-window Natures follows directly from Definition 5.5.

Lemma 5.5 (Composition of Universal Reductions for Time-Evolving k-Window Natures)**.** *Let C_1, C_2, C_3 be security games, let $\mu : \mathbb{N} \to [0,1]$, and let k be a polynomial function $k : \mathbb{N} \to \mathbb{N}$. Suppose there exists an ϵ_1-universal reduction from C_2 to C_1 w.r.t. time-evolving (k,μ)-window*

Natures, and an ϵ_2-universal reduction from C_3 to C_2 w.r.t. time-evolving (k,μ)-window Natures. Then, there exists an $\epsilon^\star$-universal reduction from C_3 to C_1 w.r.t. time-evolving (k,μ)-window Natures, where $\epsilon^\star(\lambda,a)=\epsilon_1(\lambda,\epsilon_2(\lambda,a))$ for all $\lambda\in\mathbb{N}$ and $a\in[0,1]$.

Proof. Let Nat be any time-evolving (k,μ)-window Nature. Let A_3 be any PPT attacker, and denote $a(\cdot)$ the robust advantage of (A_3,Nat) for C_3. Since there is a ϵ_2-reduction from C_3 to C_2 w.r.t. time-evolving (k,μ)-window Natures, then there exists PPT A_2 s.t. (A_2,Nat) has robust advantage $\epsilon_2(\cdot,a(\cdot))$ for C_2. Since there is a ϵ_1-reduction from C_2 to C_1 w.r.t. time-evolving (k,μ)-window Natures, then there must exist PPT A_1 s.t. (A_1,Nat) has robust advantage $\epsilon_1(\cdot,\epsilon_2(\cdot,a(\cdot)))$ for C_1. □

On Dummy Adversaries. The Dummy Lemma (see Theorem 3.3) does not appear to hold for time-evolving k-window Natures, for the same reason it does not hold for k-window Natures. See Section 5.1 for details.

5.3 Universal Reductions from Classical Non-adaptive Reductions

This section presents the main result for time-evolving k-window Natures. We show that classical *non-adaptive* straightline black-box reductions imply universal reductions w.r.t. time-evolving k-window Natures, for any choice of polynomial $k(\cdot)$. The consequence is a natural interpretation of the power of a non-adaptive straightline black-box reductions.

First, we formalize the notion of a *non-adaptive* reduction. At a high level, a non-adaptive reduction is one where the reduction decides ahead of time how to interact with the adversary, even when playing multiple security games, prior to receiving any responses from the adversary. For 1-round security games, this means that the reduction decides which queries $q_1,\ldots,q_m$ it will send ahead of time and can send all of the queries in parallel to the adversary. We extend this notion to capture general r-round security games by stipulating that the reduction initially specifies (ahead-of-time) machines $M_1,\ldots,M_m$ that individually interact with the adversary.

Definition 5.7 (Non-Adaptive Reductions)**.** Let C, C' be security games, $a\colon\mathbb{N}\to[0,1]$, $m\colon\mathbb{N}\to\mathbb{N}$, $k:\mathbb{N}\to\mathbb{N}$, $\epsilon\colon\mathbb{N}\times[0,1]\to[0,1]$. We say that a pair of uniform PPT ITMs (R_1,R_2) is an (m,ϵ)-non-adaptive straightline black-box reduction from C to C' if for every classical adversary A with advantage $a(\cdot)$ in C, the following oracle machine R^A that makes use of (R_1,R_2) has advantage $\epsilon(\cdot,a(\cdot))$ in C':

$R^A(1^\lambda)$ first simulates $R_1(1^\lambda)$ in an interaction with the challenger until R_1 outputs descriptions of oracle machines $M_1,\ldots,M_{m(\lambda)}$, each of which never restarts or rewinds their oracle. We require that $R_1(1^\lambda)$ always outputs oracle machines of this format regardless of which challenger it is interacting with. For $i\in[m(\lambda)]$, R^A subsequently runs $y_i\leftarrow M_i^A(1^\lambda)$ (using R^A's own oracle to emulate M_i's oracle access, and restarting A for each i). Finally, R^A invokes $R_2(1^\lambda,y_1,\ldots,y_{m(\lambda)})$, continuing the interaction with the challenger, running R_2 until it halts.

In the following theorem, we show that non-adaptive, straightline black box reductions for classical security games imply universal reductions for time-evolving k-window Natures.

Theorem 5.6. *Let $\epsilon\colon\mathbb{N}\times[0,1]\to[0,1]$, $m(\cdot)$ be a polynomial function, and C,C' be security games. Suppose there exists a (m,ϵ)-non-adaptive straightline black-box reduction from C to C'. Then for any polynomial $p(\cdot)$ and $k(\cdot)$, and any $\mu:\mathbb{N}\to[0,1]$, there exists a $\epsilon^\star$-universal reduction from C*

to C' w.r.t. time-evolving (k, μ)-window Natures, where $\epsilon^\star(\lambda, a) = \epsilon(\lambda, a) - 1/p(\lambda) - m(\lambda) \cdot \mu(\lambda)$ for all choices of $\lambda \in \mathbb{N}, a \in [0, 1]$.

Proof of Theorem 5.6. Let $k(\cdot)$, $m(\cdot)$ and $p(\cdot)$ be any polynomial functions (as specified in the statement of the Theorem).

The construction. Let A be any PPT attacker. Let $r(\cdot)$ be the function s.t. in any execution, $A(1^\lambda)$ makes at most $r(\lambda)$ queries to Nat. Since A is PPT, $r(\cdot)$ is a polynomial. We construct a new universal reduction A' from C to C', where A' depends on A (and in particular, on $r(\cdot)$). The goal is to later show that this universal reduction A' works for all time-evolving (k, μ)-window Natures.

By assumption, there exists a (m, ϵ)-non-adaptive classical straightline black-box reduction from C to C'—denote (R_1, R_2). Let A_{reset} be the attacker that on input 1^λ first sends $k(\lambda)$ dummy $\perp$ queries to Nature, and then runs $A(1^\lambda)$ (as defined in Lemma 5.4). A' then behaves as follows, on input 1^λ and in an interaction with C' and any Nature Nat. Let $m^*(\lambda) = m(\lambda)^2 \cdot p(\lambda)$, and let $m = m(\lambda), m^* = m^*(\lambda), p = p(\lambda), r = r(\lambda), k = k(\lambda)$ for ease of notation.

- A' simulates $R_1(1^\lambda)$, forwarding communication between C' and R_1, until R_1 halts and outputs oracle machines $M_1, \ldots, M_m$.
- A' then runs each machine $M_1, \ldots, M_m$ in an interaction with Nat according to the following procedure. Let $M_{m+1}, \ldots, M_{m^*}$ be copies of M_1. A' first samples a random permutation $\pi : [m^*] \to [m^*]$, and defines $M_i^* = M_{\pi^{-1}(i)}$ for each $i \in [m^*]$. (That is, M_i^* is the ith machine in the list of randomly permuted machines.)

  Now, for each $i \in [m^*]$ in incrementing order, starting with $i = 1$, A' runs $M_i^*(1^\lambda)$ in an interaction with a fresh copy of $A_{\mathsf{reset}}(1^\lambda)$ until M_i^* halts and outputs a string y_i^*. Importantly, A' communicates with Nat on behalf of A_{reset}, sending A_{reset}'s queries to Nat, and forwarding Nat's reply back to A_{reset}. More formally, for each i, starting with $i = 1$:

  - A' starts the ith iteration (also known as the ith *session*) by instantiating an new instance of $M_i^*(1^\lambda)$ and a new fresh instance of $A_{\mathsf{reset}}(1^\lambda)$. A' routes A_{reset}'s queries to Nat and forwards back the corresponding replies. We stress that for each iteration i, A' starts a new copy of A_{reset}; thus in every single iteration, A' will send Nature $k(\lambda)$ dummy $\perp$ messages on behalf of a new copy of A_{reset} corresponding to that iteration only.
  - Each time M_i^* makes an oracle query, A' forwards the query to A_{reset}, and sends the corresponding reply from A_{reset} back to M_i^*. A' continues the simulated interaction until M_i^* halts.
  - Before finishing iteration i, A' ensures that it has sent exactly $i \cdot (r + k)$ messages to Nat, starting from the beginning of the execution up to this point. If at this point A' has sent fewer than $i \cdot (r + k)$ messages (that is, if the simulation of A_{reset} generated fewer than $r + k$ messages) A' now sends "dummy" $\perp$ messages to Nat until A' has sent exactly $i \cdot (r + k)$ messages to Nat. Note that these $\perp$ messages do not affect the output y_i of M_i^*, as it has already been determined. Only when A' has sent $i \cdot (r + k)$ messages in total to Nat does A' end session i and proceed to session $i + 1$ (to run M_{i+1}^*).

    Remark 5.1. By adding padding messages, we ensure that for every choice of permutation π, the number of messages that Nat has seen before interacting with M_i^* for each i for the first time (via the session-specific copy of A_{reset}) remains unchanged. That is, for

each i, Nat should have received exactly $(i-1)\cdot(r+k)+k$ messages before interacting with M_i^* for the first time, independently of the choice of π. Finally, we note that A_{reset} sends exactly k dummy messages to Nat at the beginning of each session/iteration. This is intentional and important; we shall see later that the choice of k allows us to essentially flush the memory of a (time-evolving) k-window Nature at the beginning of each session, so that its subsequent behavior is independent of prior sessions.

- Let $y_i := y^*_{\pi(i)}$ for each $i \in [m]$ (noting that $M_i = M^*_{\pi(i)}$); unused outputs y^*_j for $j \notin \pi([m])$ are discarded. Each y_i corresponds to the output of machine M_i. A' now runs $R_2(1^\lambda, y_1, \ldots, y_m)$ in an interaction with C' (or an observer O) until C' (or O) halts.

Note that the number of queries A' makes to Nat scales linearly with the choice of $m^*(\lambda) = m(\lambda)^2 \cdot p(\lambda)$. Because $k(\cdot)$ is also a polynomial function, then A' makes at most a polynomial number of queries, qualifying as an attacker (which is PPT in our model).

Overview of the correctness argument. Let A be any PPT attacker, and denote A' to be the attacker constructed above. Suppose Nat is an arbitrary time-evolving (k,μ)-window Nature. Letting $a(\cdot)$ denote (A, Nat)'s robust advantage for C, we want to show that (A', Nat) has advantage $\epsilon^*(\cdot, a(\cdot))$ for C', where $\epsilon^*(\lambda, a(\lambda)) = \epsilon(\lambda, a(\lambda)) - 1/p(\lambda) - m(\lambda)\cdot\mu(\lambda)$ for all $\lambda \in \mathbb{N}$. Then applying Lemma 5.3 gives us a "full" ϵ^*-universal reduction from C to C' w.r.t. time-evolving (k,μ)-window Natures.

Torwards showing the above, we first use (A, Nat) and Nat's time-evolving (k,μ)-window property to construct a classical adversary A_{classic} that mimics (A, Nat)'s behavior in a special useful way described below. Next, we show that A_{classic} has advantage $a(\cdot)$ for C. By the assumption that there is an ϵ-non-adaptive black-box reduction $R = (R_1, R_2)$ from C to C', it follows that $R^{A_{\mathsf{classic}}}$ has advantage $\epsilon(\cdot, a(\cdot))$ for C'. We next show, through a hybrid argument, that for all $\lambda \in \mathbb{N}$

$$\Delta(\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)], \mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]) \le 1/p(\lambda) + m(\lambda)\cdot\mu(\lambda)$$

and thus (A', Nat) has advantage $\epsilon^*(\lambda, a(\lambda)) = \epsilon(\lambda, a(\lambda)) - 1/p(\lambda) - m(\lambda)\cdot\mu(\lambda)$ for C' on all $\lambda \in \mathbb{N}$, concluding the proof.

The classical adversary A_{classic}. We use (A, Nat) to construct a classical adversary A_{classic} with advantage $a(\cdot)$ for C.

First, some notation for convenience of the remainder of the proof. Let $m^* = m^*(\lambda)$ and $r = r(\lambda)$ and $k = k(\lambda)$ for ease of notation when λ is clear. For any $x \in \mathbb{N}$, denote $\rho^\perp_x$ to be the prefix of x number of dummy messages and 0 randomness, namely $(0, \perp, \perp, \ldots)$ of length x. Define the following subroutine $S(1^\lambda, i)$ that takes as input an index i; in an interaction with the challenger, $S(1^\lambda, i)$ does the following:

1. S internally runs $\mathsf{Nat}(1^\lambda, \rho^\perp_{(i-1)\cdot(k+r)})$ and $A_{\mathsf{reset}}(1^\lambda)$. Recall that A_{reset} essentially runs $A(1^\lambda)$, but first sends k dummy messages to Nature; see Lemma 5.4.

   Whenever A_{reset} sends a query to Nature, S directs it to its copy of Nat (which has been prepared as above), and routes the reply back to A_{reset}. On receiving a query from the challenger, S sends it to A_{reset}, and replies with A_{reset}'s reply.

We are ready to define A_{classic}. $A_{\mathsf{classic}}(1^\lambda)$ first samples a random $i \leftarrow [m^*]$, and then runs $S(1^\lambda, i)$ to respond to queries for the remainder of the interaction. By definition, then, for any O

$$\mathsf{out}_O[\langle O \leftrightarrow A_{\mathsf{classic}}\rangle(1^\lambda)] \equiv \mathsf{out}_O[i \leftarrow [m^*] : \langle O \leftrightarrow S(i)\rangle(1^\lambda)] \tag{7}$$

$$\equiv \mathsf{out}_O[i \leftarrow [m^*] : \langle O \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^{\perp}_{(i-1)\cdot(k+r)})\rangle(1^\lambda)]. \tag{8}$$

To motivate the above construction, roughly speaking, imagine an interaction with Nat split into m^* sessions, one for each security game, and each of length $(k+r)$ number of queries. Notably, in this imaginary interaction, each of the m^* sessions contains k "dummy" queries at the beginning of the session that "resets" the behavior of Nat because Nat is a (time-evolving) k-window Nature. In our construction, A_{classic} essentially samples one of these m^* sessions uniformly at random to play its own security game, in a way that mimics Nat's behavior even if the "prior" sessions in reality comprise queries that are not comprised of $\perp$, as we will later show.

Claim 5.7. *A_{classic} has (classical) advantage $a(\cdot)$ for C.*

Proof. Fix any $\lambda \in \mathbb{N}$. Then, denoting Q the set of all prefixes comprising $k = k(\lambda)$ dummy queries (and any randomness),

$$\begin{aligned}
\Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{classic}}\rangle(1^\lambda)] = 1\right] &\geq \min_{i\in[m^*]} \Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^{\perp}_{(i-1)\cdot(r+k)})\rangle(1^\lambda)] = 1\right] \\
&\geq \min_{i\in[m^*],\rho\in Q} \Pr\left[\mathsf{out}_C[\langle C \leftrightarrow A \leftrightarrow \mathsf{Nat}(\rho^{\perp}_{(i-1)\cdot(r+k)} \circ \rho)\rangle(1^\lambda)] = 1\right] \\
&\geq a(\lambda)
\end{aligned}$$

where the last inequality follows from the robust advantage of (A, Nat) for C. □

Showing the reduction simulates $R^{A_{\mathsf{classic}}}$. We continue to show that for all $\lambda \in \mathbb{N}$, $\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)]$ and $\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$ are $(1/p(\lambda) + m(\lambda)\cdot\mu(\lambda))$-close.

Claim 5.8. *For all $\lambda \in \mathbb{N}$, we have:*

$$\Delta(\mathit{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)], \mathit{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]) \leq \frac{1}{p(\lambda)} + m(\lambda)\cdot\mu(\lambda)$$

Proof. Again, when context is clear, denote $m = m(\lambda), m^* = m(\lambda)$, and $r = r(\lambda)$. Recall that R is the oracle machine that on input 1^λ first runs R_1 to generate oracle machines $M_1, \ldots, M_m$, and then uses its oracle to compute outputs $y_1, \ldots, y_m$ for $M_1, \ldots, M_m$ (simulating their respective oracles using R's own oracle), and then finally runs R_2 on $y_1, \ldots, y_m$. We consider an additional hybrid experiment $\mathsf{out}_{C'}[\langle C' \leftrightarrow R'\rangle(1^\lambda)]$, where R' (on input 1^λ) as before first runs R_1 in an interaction with the challenger to generate machines $M_1, \ldots, M_m$. However, to generate $y_1, \ldots, y_m$, R' now samples a random *permutation* $\pi : [m^*] \to [m^*]$, and then for each $i \in [m]$, R' uses $S(1^\lambda, \pi(i))$ to respond to oracle queries from M_i. Finally, as before, R' invokes R_2 on $y_1, \ldots, y_m$ to conclude the interaction.

We next show that for all $\lambda \in \mathbb{N}$,

$$\Delta(\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)], \mathsf{out}_{C'}[\langle C' \leftrightarrow R'\rangle(1^\lambda)]) \leq \frac{1}{p(\lambda)}.$$

Intuitively, to emulate the oracle for a machine M_i, both $R^{A_{\mathsf{classic}}}$ and R' run an instance of $S(1^\lambda, i^*)$, where in the first experiment, i^* is chosen randomly from $[m^*]$, and in the second, i^* is chosen randomly from $[m^*]$ conditioned on no i^* being chosen twice for multiple machines $M_i \neq M_{i'}, i, i' \in [m]$. If we take m^* to be big enough, the probability of collision in the first experiment can be made small. We proceed to the formal statement and its proof:

Subclaim 5.9. $\Delta(\mathit{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)], \mathit{out}_{C'}[\langle C' \leftrightarrow R'\rangle(1^\lambda)]) \leq \frac{1}{p(\lambda)}$.

Proof. Denote $p = p(\lambda), m = m(\lambda), m^* = m(\lambda)$, and $r = r(\lambda)$. Recall that each y_i in the first experiment is generated using $S(1^\lambda, i^*)$ for a random $i^* \leftarrow [m^*]$. Let COL be the event that two views $y_i, y_{i'}$ s.t. $i \neq i'$, $i, i' \in [m]$ are generated using the same i^*. As the index i^* is chosen randomly, it follows that any individual pair will collide with probability at most $1/m^*$. By a union bound, it follows that

$$\Pr[\mathsf{COL}] \leq \binom{m}{2} \cdot \frac{1}{m^*} = \binom{m}{2} \cdot \frac{1}{m^2 p} \leq \frac{1}{p}.$$

Conditioning on the event that COL does not occur, then the two experiments are identical. □

Next, we show that,

Subclaim 5.10. *For all $\lambda \in \mathbb{N}$,*

$$\Delta(\mathit{out}_{C'}[\langle C' \leftrightarrow R'\rangle(1^\lambda)], \mathit{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]) \leq m(\lambda) \cdot \mu(\lambda).$$

Proof. Fix λ, and ρ. Denote $k = k(\lambda), m = m(\lambda), m^* = m(\lambda)$, and $r = r(\lambda)$. We show the claim by a hybrid argument. Recall that A' starts with a sequence of machines $M_1, \ldots, M_{m^*}$ (of which only the first $m < m^*$ are "relevant"), and for each machine $i \in [m^*]$ runs an interaction $M_i \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}$, using A''s own access to Nat (which it cannot restart), and where each M_i and A_{reset} is simulated by A', and moreover each M_i interacts with a fresh instance of A_{reset} (thus for each M_i, A' sends k new dummy messages to Nat on behalf of A_{reset}). Moreover recall that A' samples a random permutation $\pi : [m^*] \to [m^*]$ that determines the order in which it runs the interactions, namely letting $M^*_{\pi(i)} = M_i$, the interactions are run in order $(M^*_1 \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}), \ldots, (M^*_{m^*} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat})$.

We call an interaction (indexed by j) 'relevant' if $j = \pi(i)$ for some $i \in [m]$. In other words, the outcome of a 'relevant' interaction in an execution is not discarded, is mapped to one of the original machines in $M_1, \ldots, M_m$, and is ultimately fed as input into R_2.

Now, for each $i \in [m]$, consider the attacker A'_i that runs A' until A' finishes running a 'relevant' interaction for the ith time. Subsequently, A'_i stops talking to Nat, and instead runs $S(1^\lambda, \pi(j))$ to reply to future machines M^*_j in lieu of using Nat. Observe that

$$\mathsf{out}_{C'}[\langle C' \leftrightarrow R'\rangle(1^\lambda)] \equiv \mathsf{out}_{C'}[\langle C' \leftrightarrow A'_0 \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$$

and

$$\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] \equiv \mathsf{out}_{C'}[\langle C' \leftrightarrow A'_m \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)].$$

We now show that for all $i \in [m]$

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_{C'}[\langle C' \leftrightarrow A'_i \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)], \\ \mathsf{out}_{C'}[\langle C' \leftrightarrow A'_{i+1} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda), \tag{9}$$

and then the statement of the claim follows directly by hybrid argument.

Let x be any execution in $\mathsf{Supp}(\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda))$, but truncated at the point A' finishes running a "relevant" interaction for the ith time (that is, x could also be a prefix of an execution using A'_i instead of A'). Suppose that in this truncated execution A' has finished running j total sessions (i of which are relevant). We are interested in the $j+1$th session. Consider M^*_{j+1} specified by x (i.e. the machine that is "next up" at the point of truncation—note that x fixes j and the machines and order in which they are run $M^*_1, \ldots, M^*_m$, because A' chooses them ahead of time). Let ρ^x be all of the messages and coins received by Nat in x, during the first j sessions.

Suppose for the sake of contradiction that there is a distinguisher D that distinguishes the two experiments in Equation 9 with probability μ. Then we show in Subclaim 5.11 that for some choice of x (depending on λ)

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)], \\ \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow S(j+1)\rangle(1^\lambda)] \end{pmatrix} > \mu(\lambda). \tag{10}$$

It remains to show that Equation 10 contradicts the time-evolving (k, μ)-window property of Nat (Definition 5.4). By the construction of S,

$$\mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow S(j+1)\rangle(1^\lambda)] \equiv \mathsf{out}_C[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^\perp_{j\cdot(k+r)})\rangle(1^\lambda)]$$

which implies

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)], \\ \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^\perp_{j\cdot(k+r)})\rangle(1^\lambda)] \end{pmatrix} > \mu(\lambda).$$

Now observe that $\|\rho^x\| = \left\|\rho^\perp_{j\cdot(k+r)}\right\| = j \cdot (k+r)$, and thus we contradict Lemma 5.4, concluding the proof. □

Subclaim 5.11. *Fix any $\lambda \in \mathbb{N}$, and suppose that there is a distinguisher D that distinguishes the two experiments in Equation 9 with probability μ. Then there exists some truncated execution x s.t.*

$$\bigtriangleup \begin{pmatrix} \mathit{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathit{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)], \\ \mathit{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow S(j+1)\rangle(1^\lambda)] \end{pmatrix} > \mu(\lambda). \tag{11}$$

Proof. For any truncated execution x, define a distinguisher machine D^x as follows:

- D^x gets as input either $y \leftarrow \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)]$ or $y \leftarrow \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow S(j+1)\rangle(1^\lambda)]$.
- D^x now samples an execution $z \leftarrow \langle C' \leftrightarrow A'_i \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)$ conditioned on x being a prefix of z, and y being equal to the output of M^*_{j+1} in z. (This can be done by running a continuation of x, which is possible since A'_i no longer interacts with Nat after finishing the jth session, so we can condition on y as the output of the $j+1$th session without worrying about updating the state of Nat for future sessions.)
- Finally, D^x outputs $D(\mathsf{out}_{C'}[z])$.

It remains to show that for some choice of x, D^x has advantage μ in distinguishing its input. To help, we define an experiment Exp that on input $b \in \{0, 1\}$:

- First, samples a random x (by running a random execution of $\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)$ until A' has finished i relevant interactions),
- samples either (if $b = 0$) $y \leftarrow \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)]$, or (if $b = 1$) $y \leftarrow \mathsf{out}_{M^*_{j+1}}[\langle M^*_{j+1} \leftrightarrow S(j+1)\rangle(1^\lambda)]$,
- and finally outputs $D_x(y)$.

Observe that Exp in fact invokes D on exactly the distribution $\mathsf{out}_{C'}[\langle C' \leftrightarrow A'_i \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$ if $b = 1$, and $\mathsf{out}_{C'}[\langle C' \leftrightarrow A'_{i+1} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$ if $b = 0$. Thus, by the assumption that D is a good distinguisher, $\Pr[\mathsf{Exp}(1) = 1] - \Pr[\mathsf{Exp}(0) = 1] > \mu$. Denote X the random variable for the value of x sampled by Exp. We thus have:

$$\sum_x \Pr[X = x] \cdot \Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \sum_x \Pr[X = x] \cdot \Pr[\mathsf{Exp}(0) = 1 \mid X = x] > \mu(\lambda)$$

which simplifies to

$$\sum_x \Pr[X = x] \cdot (\Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \Pr[\mathsf{Exp}(0) = 1 \mid X = x]) > \mu(\lambda)$$

Thus there must exist a choice of x s.t.

$$\Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \Pr[\mathsf{Exp}(0) = 1 \mid X = x] > \mu(\lambda)$$

immediately implying Equation 10 for that choice of x, concluding the proof. □

Finally, combining Claim 5.9 and Claim 5.10, by triangle inequality, $\forall \lambda \in \mathbb{N}$,

$$\Delta(\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)], \mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]) \leq \frac{1}{p(\lambda)} + m(\lambda) \cdot \mu(\lambda)$$

as required. □

Concluding the success probability analysis. By the correctness of R, and Claim 5.7, we have that $R^{A_{\mathsf{classic}}}$ has advantage $\epsilon(\cdot, a(\cdot))$ for C'. Combining this with Claim 5.8, we thus have for all $\lambda \in \mathbb{N}$

$$\begin{aligned}
\Pr\big[\mathsf{out}_{C'}[\langle C' \leftrightarrow A' \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] = 1\big] &\geq \Pr\big[\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{A_{\mathsf{classic}}}\rangle(1^\lambda)] = 1\big] - \left(\frac{1}{p(\lambda)} + m(\lambda) \cdot \mu(\lambda)\right) \\
&\geq \epsilon(\lambda, \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{classic}}\rangle(1^\lambda)] = 1\big]) - \left(\frac{1}{p(\lambda)} + m(\lambda) \cdot \mu(\lambda)\right) \\
&\geq \epsilon(\lambda, a(\lambda)) - \left(\frac{1}{p(\lambda)} + m(\lambda) \cdot \mu(\lambda)\right)
\end{aligned}$$

as desired. □

6 Universal Reductions imply Standard Reductions

We show that if there is a universal reduction with respect to k-window Natures, where k is large enough to accommodate the security g'ame in question, then there is a classical reduction with respect to most standard classes of attackers. Importantly, any "plain" universal reduction implies a universal reduction w.r.t. (k, μ)-window Natures for any choice of k and μ, so this is the strongest notion.

Theorem 6.1. *Let $\mu : \mathbb{N} \to [0, 1]$ be an arbitrary function. Let C and C' be security games, and let $k(\cdot)$ be a polynomial function that upper bounds the number of rounds in any interaction with C. Assume there exists a ϵ-universal reduction from C to C' w.r.t. (k, μ)-window Natures. Then there exists a ϵ-reduction w.r.t. PPT, non-uniform PPT, QPT, and non-uniform QPT attackers.*

We first prove this theorem for the case of PPT attackers. We then discuss how this directly generalizes to non-uniform PPT and QPT attackers. Then, we prove it for the case of non-uniform QPT attackers to highlight the different techniques required in this case.

Lemma 6.2. *Let $\mu : \mathbb{N} \to [0, 1]$ be an arbitrary function. Let C and C' be security games, and let $k(\cdot)$ be a polynomial function that upper bounds the number of rounds in any interaction with C. Suppose that there exists a ϵ-universal reduction from C to C' w.r.t. (k, μ)-window Natures. Then there exists a ϵ-reduction w.r.t. PPT attackers.*

Proof. Let B be a PPT attacker such that for all $\lambda \in \mathbb{N}$, B wins security game C with probability $a(\lambda)$. Let $k(\cdot)$ be the function that upper bounds the number of queries that C might send. Consider the dummy attacker A_{dummy} and let Nat_B be the Nature that, on input 1^λ and on receiving a new $\mathsf{startsession}$ message, starts a ticker $i = 0$, and internally runs an instance of B to reply to queries, incrementing i each time after it responds to a query. If at any point $i = k(\lambda)$, Nat_B stops running its instance of B, and responds with $\perp$ to all future queries until the next time it receives a $\mathsf{startsession}$ message, at which point it repeats the above procedure all over again, resetting $i = 0$.

We note that: (1) Nat_B is a (k, μ)-window Nature (for any choice of μ), and (2) for any prefix $\rho \in \{0, 1\}^*$, $(A_{\mathsf{dummy}}, \mathsf{Nat}_B(\rho))$ wins the security game C with the same probability $a(\lambda)$ as B. The latter follows because the prefix ρ has no effect on Nat_B since A_{dummy} immediately sends Nat_B a $\mathsf{startsession}$ at the beginning of the interaction. Thus, by applying the universal reduction for (k, μ)-window Natures that exists by assumption, there exists a PPT attacker A' such that (A', Nat_B) wins the security game C' with probability $\epsilon(\lambda, a(\lambda))$ on security parameter λ.

Given (A', Nat_B), we construct a PPT attacker B' as follows. B' receives messages for C' and emulates the PPT machine A' and Nat_B to respond to those messages. To simulate Nat_B, note that B' will invoke the code of B (for up to $k(\lambda)$ rounds each time). As A' and Nat_B are PPT ITMs, their composition is also a PPT ITM, so B' is a PPT machine. Furthermore, B' perfectly emulates (A', Nat_B), so B' succeeds with probability $\epsilon(\lambda, a(\lambda))$, as required. □

The above proof outline immediately generalizes to non-uniform PPT attackers. The only difference is that we need the Nature Nat_B to hardcode the non-uniform advice of B. The simulation of B' will also be non-uniform PPT since B' will also need to hardcode the advice of B.

For the case of uniform QPT attackers, we note that such attackers have a uniform, classical description, so the code of B can be copied and restarted without issue. To actually perform the computation, Nat_B needs to simulate any quantum operations that the QPT machine B specifies, but it can do this to arbitrary precision in an exponential amount of time (see e.g. [BV97, NC16]

for details). Then, B' can run these computations in QPT while simulating A' and the potentially many copies of B, as required.

To deal with non-uniform QPT attackers, we require just a little bit of extra work: Essentially, we require B' to hard-code *many* copies of the non-uniform quantum advice of the original attacker B; this is needed in order to simulate restarts of B:

Lemma 6.3. *Let $\mu : \mathbb{N} \to [0,1]$ be an arbitrary function. Let C and C' be security games, and let $k(\cdot)$ be a polynomial function that upper bounds the number of rounds in any interaction with C. Assume there exists a ϵ-universal reduction from C to C' w.r.t. (k,μ)-window Natures. Then there exists a ϵ-reduction w.r.t. non-uniform QPT attackers.*

Proof. Now, let B be a non-uniform QPT attacker, meaning B is a quantum polynomial time machine with polynomially bounded quantum advice, such that for all $\lambda \in \mathbb{N}$, B wins security game C on input 1^λ with probability $a(\lambda)$. Let $k(\cdot)$ be the function that upper bounds the number of queries that C might send. Consider the dummy attacker A_{dummy} and let Nat_B be the Nature that simply runs a fresh copy of the code of B with the appropriate advice when it receives a new $\mathsf{startsession}$ message from A_{dummy}, for at most $k(\lambda)$ rounds of interaction (and after $k(\lambda)$ rounds of interaction, before it receives a subsequent $\mathsf{startsession}$ message, it just responds to every query with $\perp$). As Nat_B is allowed to run in unbounded time with arbitrarily long non-uniform advice, it can simulate many copies of B by encoding the polynomially many quantum bits of B's non-uniform advice using exponentially many classical bits of non-uniform advice (see [NC16] for further details).

We note that Nat_B is a (k,μ)-window Nature, for any choice of μ. Furthermore, since B wins security game C with probability $a(\lambda)$, so does $(A_{\mathsf{dummy}}, \mathsf{Nat}_B)$; moreover, $(A_{\mathsf{dummy}}, \mathsf{Nat}_B)$ does so robustly, as any prefix of interaction that Nat_B may have seen is ignored as soon as A_{dummy} sends its first message $\mathsf{startsession}$. Thus, by assumption, there exists a PPT machine A' such that (A', Nat_B) wins the security game C' with probability $\epsilon(\lambda, a(\lambda))$ on security parameter λ.

Given (A', Nat_B), we define the non-uniform QPT attacker B' as follows. For each $\lambda \in \mathbb{N}$, B' hardcodes an upper bound $b(\lambda)$ on the number of times that A' sends a $\mathsf{startsession}$ message to Nat_B in an execution on input 1^λ. B' then hardcodes $b(\lambda)$ copies of B, including $b(\lambda)$ copies of its non-uniform quantum advice. B' accepts messages from the game C', and responds to the messages by simulating an interaction between A' and its copies of B, invoking a fresh copy of B for each session of C that A' starts (when A' sends a $\mathsf{startsession}$ message), running each copy of B to respond to at most $k(\lambda)$ messages (and then replying $\perp$ afterwards).

To finish the proof of the lemma, it remains to show that B' has the correct success probability and is a non-uniform QPT machine. Since B' perfectly simulates (A', Nat_B) by construction (given that $b(\lambda)$ is chosen large enough), it follows that B' wins for C' with the same probability $\epsilon(\lambda, a(\lambda))$ as (A', Nat_B), as required. Next, it holds that B' is a non-uniform QPT machine since non-uniform QPT computation is closed under polynomial repetition with itself and under communication with a PPT machine. □

7 References

[AC02] Mark Adcock and Richard Cleve. A quantum goldreich-levin theorem with cryptographic applications. In *STACS*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: the hardness of quantum rewinding. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 474–483. IEEE, 2014.

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 296–315. Springer, 2013.

[BBK22] Nir Bitansky, Zvika Brakerski, and Yael Tauman Kalai. Constructive post-quantum reductions. *IACR Cryptol. ePrint Arch.*, page 298, 2022.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, 2011.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BS20] Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 269–279, 2020.

[BV97] Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper B Nielsen. Relaxing chosen-ciphertext security. In *Annual International Cryptology Conference*, pages 565–582. Springer, 2003.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: Breaking the quantum rewinding barrier. In *FOCS*, pages 49–58. IEEE, 2021.

[Die82] DGBJ Dieks. Communication by epr devices. *Physics Letters A*, 92(6):271–272, 1982.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, 1990.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, 1989.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol07] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

[Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[Gro04] Jens Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In *Theory of Cryptography Conference*, pages 152–170. Springer, 2004.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hof11] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *Journal of Cryptology*, 24(3):470–516, 2011.

[HP15] Joseph Y Halpern and Rafael Pass. Algorithmic rationality: Game theory with costly computation. *Journal of Economic Theory*, 156:246–268, 2015.

[IR95] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61, 1989 [1995].

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, 1979.

[LTW05] Henry Lin, Luca Trevisan, and Hoeteck Wee. On hardness amplification of one-way functions. In *Theory of Cryptography Conference*, pages 34–49. Springer, 2005.

[Mah18] Urmila Mahadev. Classical verification of quantum computations. In *FOCS*, pages 259–267. IEEE Computer Society, 2018.

[Mau02] Ueli Maurer. Indistinguishability of random systems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 110–132. Springer, 2002.

[Mau11] Ueli Maurer. Constructive cryptography–a new paradigm for security definitions and proofs. In *Joint Workshop on Theory of Security and Applications*, pages 33–56. Springer, 2011.

[MP04] Ueli Maurer and Krzysztof Pietrzak. Composition of random systems: When two weak make one strong. In *Theory of Cryptography Conference*, pages 410–427. Springer, 2004.

[MPR07] Ueli Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. In *Annual International Cryptology Conference*, pages 130–149. Springer, 2007.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *In Innovations in Computer Science*. Citeseer, 2011.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptol.*, 4(2):151–158, 1991.

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[Pop05] Karl Popper. *The logic of scientific discovery*. Routledge, 2005.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable rcca encryption. In *Annual International Cryptology Conference*, pages 517–534. Springer, 2007.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In *International Conference on Cryptology in Vietnam*, pages 211–228. Springer, 2006.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 135–152. Springer, 2012.

[Unr16] Dominique Unruh. Computationally binding quantum commitments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 497–527. Springer, 2016.

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009.

[WZ82] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *FOCS*, pages 679–687. IEEE Computer Society, 2012.

A More on k-Window Natures

We here provide some basic observations about universal reductions w.r.t. k-window Natures.

A.1 Composition and More

We observe that the composition lemma still holds trivially w.r.t. k-window Natures.

Lemma A.1 (Composition of Universal Reductions for k-Window Natures)**.** *Let C_1, C_2, C_3 be security games, let $k(\cdot)$ be a polynomial function, and let $\mu : \mathbb{N} \to [0,1]$. Suppose there exists an ϵ_1-universal reduction from C_2 to C_1 w.r.t. (k,μ)-window Natures, and an ϵ_2-universal reduction from C_3 to C_2 w.r.t. (k,μ)-window Natures. Then, there exists an $\epsilon^\star$-universal reduction from C_3 to C_1 w.r.t. (k,μ)-window Natures, where $\epsilon^\star(\lambda, a) = \epsilon_1(\lambda, \epsilon_2(\lambda, a))$ for all $\lambda \in \mathbb{N}$ and $a \in [0,1]$.*

Proof. Let (A_3, Nat) be any (k,μ)-window Nature, and denote $a(\cdot)$ its robust advantage in C_3. Since there is a ϵ_2-reduction from C_3 to C_2 w.r.t. (k,μ)-window Natures, then there exists PPT A_2 s.t. (A_2, Nat) has robust advantage $\epsilon_2(\cdot, a(\cdot))$ for C_2. Since there is a ϵ_1-reduction from C_2 to C_1 w.r.t. (k,μ)-window Nature, then there must exist PPT A_1 s.t. (A_1, Nat) has robust advantage $\epsilon_1(\cdot, \epsilon_2(\cdot, a(\cdot)))$ for C_1. □

Win-once universal reductions also continue to imply regular universal reductions w.r.t. k-window Natures:

Lemma A.2 (Win-once Universal Reductions imply Universal Reductions for k-Window Natures)**.** *Let $k(\cdot)$ be a polynomial function, and let $\mu : \mathbb{N} \to [0,1]$, ϵ: $\mathbb{N} \times [0,1] \to [0,1]$, and let C and C' be security games. If there exists a* win-once ϵ-universal reduction *from C to C'* w.r.t. (k,μ)-window Natures*, then there exists a* ϵ-universal reduction *from C to C'* w.r.t. (k,μ)-window Natures*.*

Proof. The proof is identical to that for the time-evolving setting (see Lemma 5.3), except deleting the words 'time-evolving' throughout, and removing the $\|\rho'\| = \|\rho''\|$ constraint in Equation 6. Equation 6 then directly follows from the (k,μ)-window property of Nat (which is stronger than what we assume in Lemma 5.3, as necessary to compensate for the removal of the $\|\rho'\| = \|\rho''\|$ constraint), thus showing that for any prefix ρ, Nat_ρ is a (k,μ)-window Nature, as required. □

A.2 Universal Reductions from Sequential Straightline Reductions

We next show that any sequential straightline black-box reductions yields a universal reduction w.r.t. k-window Natures.

Lemma A.3. *Let $\epsilon : \mathbb{N} \times [0,1] \to [0,1]$, $q : \mathbb{N} \to \mathbb{N}$, and C, C' be security games. Suppose there exists a ϵ-sequential straight-line black-box reduction from C to C', that on input 1^λ restarts its oracle at most $q(\lambda)$ times. Then for every function $\mu : \mathbb{N} \to [0,1]$, for every polynomial function $k : \mathbb{N} \to \mathbb{N}$, there is a ϵ^*-universal reduction from C to C' w.r.t. (k,μ)-window Natures, where $\epsilon^*(\lambda, a) = \epsilon(\lambda, a) - q \cdot \mu$.*

Proof. Choose any polynomial function $k : \mathbb{N} \to \mathbb{N}$ and any function $\mu : \mathbb{N} \to [0,1]$. We want to show that given an ϵ-straightline black-box reduction from C to C', we can construct a new universal reduction R' that works for any (k,μ)-window Nature, using oracle access to the attacker.

The Construction. By assumption there exists a ϵ-straightline black-box reduction R from C to C' (where R has oracle access to some adversary algorithm). Let R'^A be the universal reduction (communicating with some Nature Nat) that on input 1^λ, with oracle access to some PPT attacker A, essentially uses its oracle to A and interactive access to Nat to simulate an interaction between $R \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}$ (where A_{reset} is based on A). It does this by running internally a copy of $R(1^\lambda)$ and a copy of $A_{\mathsf{reset}}(1^\lambda)$, communicating with Nat on behalf of A_{reset}, and sending R's oracle queries to A_{reset} (forwarding the corresponding replies). Each time R restarts its oracle, R'^A starts a new copy of $A_{\mathsf{reset}}(1^\lambda)$ replacing the previous instance, communicates with Nat on behalf of the new instance of A_{reset} (thus immediately sending $k(\lambda)$ dummy messages to Nat), and sends R's future oracle queries to the new copy of A_{reset}. Since $k(\cdot)$ is a polynomial function and A is polynomial time, then R'^A is polynomial time as required of the attacker.

Correctness Argument. Let (A, Nat) be any augmented adversary such that Nat is a (k, μ)-window Nature. Denote $a(\cdot)$ the robust advantage of (A, Nat) for C. We now argue that (R'^A, Nat) has advantage $\epsilon^*(\cdot, a(\cdot))$ for C'. Applying the 'win once lemma' Lemma A.2, then (R'^A, Nat) has robust advantage $\epsilon^*(\cdot, a(\cdot))$ for C', proving the lemma.

For the purposes of the proof, define the following classical (possibly inefficient) subroutine S that interacts with a challenger (playing the role of an adversary). S on input 1^λ runs internally instances of $\mathsf{Nat}(1^\lambda)$ and $A_{\mathsf{reset}}(1^\lambda)$; whenever A_{reset} sends a query to Nature, S directs it to its copy of Nat, and routes the reply back to A_{reset}. On receiving a query from the challenger, S sends it to A_{reset}, and replies with A_{reset}'s reply. Recall that A_{reset} essentially runs $A(1^\lambda)$, but first sends $k(\lambda)$ dummy messages to Nature; see the definition in Lemma 5.1. Immediately we know that S has advantage $a(\cdot)$ in C: for all $\lambda \in \mathbb{N}$,

$$\begin{aligned}\Pr\big[\mathsf{out}_C[\langle C \leftrightarrow S\rangle(1^\lambda)] = 1\big] &= \Pr\big[\mathsf{out}_C[\langle C \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] = 1\big] \\ &\geq a(\lambda)\end{aligned}$$

where the equality follows by construction, and the inequality follows from the (robust) advantage of $(A_{\mathsf{reset}}, \mathsf{Nat})$ for C.

Consequently, by the correctness of R, since S is a classical adversary, for all $\lambda \in \mathbb{N}$,

$$\Pr\big[\mathsf{out}_{C'}[\langle C' \leftrightarrow R^S\rangle(1^\lambda)] = 1\big] \geq \epsilon(\lambda, a(\lambda)). \tag{12}$$

It remains to argue that for all $\lambda \in \mathbb{N}$,

$$\Delta(\mathsf{out}_{C'}[\langle C' \leftrightarrow R'^A \leftrightarrow \mathsf{Nat}(\rho)\rangle(1^\lambda)], \mathsf{out}_{C'}[\langle C' \leftrightarrow R^S\rangle(1^\lambda)]) \leq q(\lambda) \cdot \mu(\lambda). \tag{13}$$

We show Equation 13 by hybrid argument. Let $q(\cdot)$ be a polynomial s.t. $R(1^\lambda)$ restarts its oracle at most $q(\lambda)$ times in any execution (recall that R is PPT) for all $\lambda \in \mathbb{N}$. Now, for each $i \in [q]$, consider the reduction R'^A_i that runs R'^A until R tries to restart its oracle for the ith time. At that point, R'^A_i stops talking to Nat, and internally uses S to answer future oracles queries from R (restarting S when R restarts its oracle). We show in Claim A.4 that for all $\lambda \in \mathbb{N}$, $i \in [q(\lambda)]$

$$\bigtriangleup \begin{pmatrix} \mathsf{out}_{C'}[\langle C' \leftrightarrow R'^A_i \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)], \\ \mathsf{out}_{C'}[\langle C' \leftrightarrow R'^A_{i+1} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] \end{pmatrix} \leq \mu(\lambda)$$

implying Equation 13 by hybrid argument, as the number of hybrids is a polynomial $q(\cdot)$.

Equations 12 and 13 together imply that for all $\lambda \in \mathbb{N}$,

$$\begin{aligned}\Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow R'^{A} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)] = 1\right] &\geq \Pr\left[\mathsf{out}_{C'}[\langle C' \leftrightarrow R^{S}\rangle(1^\lambda)] = 1\right] - q(\lambda)\cdot\mu(\lambda) \\ &\geq \epsilon(\lambda, a(\lambda) - \mu(\lambda)) - q(\lambda)\cdot\mu(\lambda)\end{aligned}$$

concluding the proof. □

Claim A.4. *For all $i \in [q(\lambda)]$, for all λ,*

$$\Delta\left(\mathit{out}_{C'}[\langle C' \leftrightarrow R'^{A}_{i} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)], \mathit{out}_{C'}[\langle C' \leftrightarrow R'^{A}_{i+1} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]\right) \leq \mu(\lambda).$$

Proof. Recall that R'^{A}_{i} directly runs R'^{A} (which runs many copies of A) for i sessions of interaction with Nat, at which point R'^{A}_{i} uses S in place of A_{reset} (and Nat) in subsequent sessions. Thus, $R'^{A}_{0} = R^{S}$ (that is, it ignores Nat) and $R'^{A}_{q(\lambda)} = R'^{A}$. So the only difference between R'^{A}_{i} and R'^{A}_{i+1} is that in R'^{A}_{i}, the $(i+1)$th interaction uses S, whereas in R'^{A}_{i+1}, it now uses A_{reset} (and thus communicates with the stateful Nat).

As an overview for our proof, we essentially use Lemma 5.1 to argue that for any fixing of the first i sessions, it is indistinguishable whether the $i+1$ session uses $A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}$ or S; that is, because each session is sending Nat $k(\lambda)$ number of dummy messages which should 'reset' Nature. The claim follows by a probabilistic argument.

More formally, fix any $\lambda \in \mathbb{N}$, $i \in [q(\lambda)]$, and let x be any execution in $\mathsf{Supp}(\langle C' \leftrightarrow R'^{A}_{i} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda))$, but truncated at the point R (which is simulated by R'^{A}_{i}) restarts its oracle for the ith time. In other words, x fixes the outcome of the first i sessions of interaction. Let ρ^x be the messages and coins received by Nat in x, and let state^x denote the joint state of C' and the simulated R at the end of x. Let M^x be the machine that runs C' and R initialized in the state state^x, simulating C''s communication with R, answering R's oracle queries by communicating with some attacker machine on behalf of R, and finally halting and outputting the joint view when R next tries to restart its oracle. In other words, M^x and ρ^x formalize how we "continue" the execution after having fixed the i sessions of interaction specified in x.

Suppose for the sake of contradiction that there is a distinguisher D that distinguishes the two experiments in the statement of the claim with probability $\mu = \mu(\lambda)$ for some λ. Then we show that for some choice of x

$$\Delta\begin{pmatrix}\mathsf{out}_{M^x}[\langle M^x \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)], \\ \mathsf{out}_{M^x}[\langle M^x \leftrightarrow S\rangle(1^\lambda)]\end{pmatrix} > \mu \tag{14}$$

which immediately implies

$$\Delta\begin{pmatrix}\mathsf{out}_{M^x}[\langle M^x \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)], \\ \mathsf{out}_{M^x}[\langle M^x \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]\end{pmatrix} > \mu$$

which contradicts Lemma 5.1 (as no prefix is also a valid prefix), thus contradicting the (k,μ)-window property of Nat.

For any truncated execution x, define a distinguisher machine D^x as follows:

- D^x gets as input either $y \leftarrow \mathsf{out}_{M^x}[\langle M^x \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)]$ or $y \leftarrow \mathsf{out}_{M^x}[\langle M^x \leftrightarrow S\rangle(1^\lambda)]$.

- D^x now samples an execution $z \leftarrow \langle C' \leftrightarrow R_i'^A \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)$ conditioned on x being a prefix of z, and y being equal to the joint view of C' and R at the exact point when R restarts its oracle for the $i+1$th time. (Sampling z in this way can be done by initializing C' and R to have the view specified by y, and then running a continuation of $\langle C' \leftrightarrow R_i'^A \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)$ starting at the $i+1$th session, which is easy using S since $R_i'^A$ no longer interacts with Nat after the ith session.)
- Finally, D^x outputs $D(\mathsf{out}_O[z])$.

It remains to show that for some choice of x, D^x has advantage μ in distinguishing its input. To help, we define an experiment Exp that on input $b \in \{0,1\}$:

- First, samples a random x (by running a random execution until R restarts its oracle for the ith time),
- samples either (if $b = 0$) $y \leftarrow \mathsf{out}_{M^x}[\langle M^x \leftrightarrow A_{\mathsf{reset}} \leftrightarrow \mathsf{Nat}(\rho^x)\rangle(1^\lambda)]$, or (if $b = 1$) $y \leftarrow \mathsf{out}_{M^x}[\langle M^x \leftrightarrow S\rangle(1^\lambda)]$,
- and finally outputs $D_x(y)$.

Observe that Exp in fact invokes D on exactly the distribution $\mathsf{out}_{C'}[\langle C' \leftrightarrow R_i'^A \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$ if $b = 1$, and $\mathsf{out}_{C'}[\langle C' \leftrightarrow R_{i+1}'^A \leftrightarrow \mathsf{Nat}\rangle(1^\lambda)]$ if $b = 0$. Thus, $\Pr[\mathsf{Exp}(1) = 1] - \Pr[\mathsf{Exp}(0) = 1] > \mu(\lambda)$. Denote X the random variable for the value of x sampled by Exp:

$$\sum_x \Pr[X = x] \cdot \Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \sum_x \Pr[X = x] \cdot \Pr[\mathsf{Exp}(0) = 1 \mid X = x] > \mu(\lambda)$$

$$\sum_x \Pr[X = x] \cdot (\Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \Pr[\mathsf{Exp}(0) = 1 \mid X = x]) > \mu(\lambda)$$

Thus there must exist a choice of x s.t.

$$\Pr[\mathsf{Exp}(1) = 1 \mid X = x] - \Pr[\mathsf{Exp}(0) = 1 \mid X = x] > \mu(\lambda)$$

immediately implying Equation 14 for that choice of x, concluding the proof. □

B Universal Reductions from Classical Results

In this section, we give various examples of standard cryptographic primitives that have single-shot, straightline, black-box reductions to other primitives. We emphasize that all of the reductions we consider are *uniform* PPT reductions. By Theorem 4.1, this provides a foundation for a theory of universal reductions for many cryptographic primitives. For each primitive, we will define the syntax and security games. Then, we will provide a brief proof sketch for the classical reduction to show how it implies a universal reduction.

B.1 Pseudorandom Generators

A pseudorandom generator (PRG) G is a length-expanding function whose output is computationally indistinguishable from random on a uniform input, with syntax and security defined as follows.

Definition B.1 (PRG Syntax)**.** Let $m\colon \mathbb{N} \to \mathbb{N}$. A m-bit stretch pseudo-random generator (PRG) is a PPT computable function $G\colon \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$, $|G(x)| = m(|x|) > |x|$.

Definition B.2 (PRG Security Game)**.** Let $m\colon \mathbb{N} \to \mathbb{N}$. The challenger C for the m-bit stretch PRG security game for G interacts with an adversary A on common input 1^λ as follows. C samples a bit $b \leftarrow \{0,1\}$. If $b = 0$, C sends A a string $y \leftarrow G(U_\lambda)$. If $b = 1$, C sends A a string $y \leftarrow U_{m(\lambda)}$. C receives a bit $b^\star$ from A and outputs 1 iff $b = b^\star$.

With respect to classical non-uniform PPT adversaries, it is well known that there exist pseudo-random generators from any one-way function [HILL99]. It is currently open whether or not there is a universal reduction from PRGs to one-way functions.

However, the classical proof (see e.g. [Gol07]) that a $(\lambda+1)$-bit stretch PRG implies an $m(\lambda)$-bit stretch PRG for any m is a single-shot straightline black-box reduction. In fact, it follows by a simple hybrid argument, which falls into this setting.

For any polynomial m, we recall the construction of an $m(\lambda)$-bit stretch PRG G_m from a $(\lambda+1)$-bit stretch PRG G'. $G(x)$ first sets $y_0 = x$, computes $G'(y_0)$, and splits the output into two parts. The first $|x|$ bits, denoted y_1, are treated as a new seed, and the last bit, denoted b_1, is used for the output. It then repeats this process with y_1 as a new seed, and so on. The full construction is as follows:

$G_m(x)$:

1. Set $y_0 = x$.
2. For each $i = 1, \ldots, m(|x|)$, set $y_i, b_i = G'(y_{i-1})$.
3. Output $b_1, \ldots, b_{m(|x|)}$.

Corollary B.1. *For any polynomial m, there exists an ϵ-universal reduction from the PRG security of G_m to the PRG security of G' for $\epsilon(\lambda, a) = 1/2 + \delta/m(\lambda)$, where $\delta = a - 1/2$.*

Proof sketch. Let C and C' be the challengers in the PRG security game for G_m and G', respectively. The single-shot, straightline black-box reduction R for the game C' that makes use of an adversary A for the game C is defined as follows. The Corollary then follows immediately by Theorem 4.1.

The challenger C' for the game G' samples a random bit $b \leftarrow \{0,1\}$ and sends R a uniform string from $s \leftarrow G'(U_\lambda)$ if $b = 0$ or $s \leftarrow U_{\lambda+1}$ if $b = 1$. Parse $s \in \{0,1\}^\lambda$ as $y'||b'$ where $y' \in \{0,1\}^\lambda$ and $b' \in \{0,1\}$. R samples an index $i \leftarrow [m(\lambda)]$. For $j = 1, \ldots, i-1$, R computes $b_j \leftarrow \{0,1\}$. For $j = i$, R computes uses $y_i = y'$ and $b_i = b'$. For $j = i+1, \ldots, m(\lambda)$, R computes y_j, b_j using G' as G_m given y_{j-1}. R sends $b_1, \ldots, b_{m(\lambda)}$ to A and receives a bit $b^\star$, which it forwards to C.

It remains to analyze the advantage of R given that A has advantage $a = 1/2 + \delta$ for some $\delta \in [-1/2, 1/2]$. The analysis follows by a hybrid argument. For each $\lambda \in \mathbb{N}$ and $i \in \{0, \ldots, m(\lambda)\}$, let $H_i(1^\lambda)$ be the distribution where $b_1, \ldots, b_i$ are uniform, then a seed $y_i \in \{0,1\}^\lambda$ is sampled and $b_{i+1}, \ldots, b_{m(\lambda)}$ are computed using G' as in G_m. Let $p_i(\lambda)$ be the probability that an adversary A outputs 1 on input $s \leftarrow H_i(1^\lambda)$. Note that $H_0(1^\lambda)$ corresponds to the output of $G_m(x)$ and $H_{m(\lambda)}(1^\lambda)$ is a uniformly random string. Thus $p_{m(\lambda)} - p_0 = 2\delta$ by assumption.

For each $i \in [m(\lambda)]$, which is chosen uniformly at random, R samples from $H_i(1^\lambda)$ if the challenger chose $b = 1$ and $H_{i-1}(1^\lambda)$ if the challenger chose $b = 0$. Thus, the reduction R succeeds

with probability $p_i(\lambda)/2 + (1 - p_{i-1}(\lambda))/2$ conditioned on i being chosen. Summing over all i, this implies that R's overall success probability is

$$\sum_{i \in [m(\lambda)]} \frac{1}{m(\lambda)} \left(p_i(\lambda)/2 + (1 - p_{i-1}(\lambda))/2\right) = \frac{1}{2} + \frac{1}{2m(\lambda)}(p_{m(\lambda)}(\lambda) - p_0(\lambda)) = \frac{1}{2} + \frac{\delta}{m(\lambda)},$$

as required. □

B.2 Pseudorandom Functions

A pseudorandom function (PRF) F is an efficient, keyed function that is indistinguishable from a random function over the choice of a random key. To formalize this notion of indistinguishability, we consider a games consisting of some fixed number $r(\lambda)$ rounds and require that security holds against all $r(\lambda)$ round games.

Definition B.3 (PRF Syntax)**.** A pseudorandom function (PRF) is a function $F\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that for all $K \in \{0,1\}^\lambda$, $F(K,\cdot)\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ and is PPT computable.

Definition B.4 (r-Round PRF Security Game)**.** Let $r\colon \mathbb{N} \to \mathbb{N}$. The challenger C for the r-round PRF security game for F interacts with an adversary A on common input 1^λ as follows. C samples a bit $b \leftarrow \{0,1\}$. If $b = 0$, C samples $K \leftarrow \{0,1\}^\lambda$ and sets the function $f(\cdot) = F(K,\cdot)$. If $b = 1$, C sets the function f to be a random function from λ bits to λ bits. C receives at most $r(\lambda)$ rounds of queries of the form $x_i \in \{0,1\}^\lambda$ from A for $i \in [r(\lambda)]$. To each query, C responds with $f(x_i)$. After all queries, A sends C a bit $b^\star$ and C outputs 1 iff $b = b^\star$.

PRFs are implied by 2λ-bit stretch PRGs via the GGM construction [GGM86] with respect to non-uniform PPT adversaries. We show that this same construction and proof implies a universal reduction from PRF security to PRG security. The proof follows by a nested hybrid argument, with security loss that depends on the number of rounds.

We next recall the GGM construction of a PRF F based on a 2λ-bit PRG G. Let $K \in \{0,1\}^\lambda$ be a key for the PRF evaluation and $x = (x_1, \ldots, x_\lambda) \in \{0,1\}^\lambda$ be the input to be evaluated. F is defined as follows:

$F(K, x)$:

1. Let $v_0 = K$.
2. For $j = 1, \ldots, \lambda$, set v_j to be the first λ bits of $G(v_{j-1})$ if $x_j = 0$ and the last λ bits of $G(v_{j-1})$ if $x_j = 1$.
3. Output v_λ.

Corollary B.2. *For any polynomial r, there exists an ϵ-universal reduction from the r-round PRF security of F to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/(\lambda \cdot r(\lambda))$, where $\delta = a - 1/2$.*

Proof sketch. The proof follows by a hybrid argument, which implies a single-shot straightline black-box reduction and hence a universal reduction by Theorem 4.1. For each $\lambda \in \mathbb{N}$, we consider hybrid ITMs $H_{j,\ell}(1^\lambda)$ for $j \in \{0, \ldots, \lambda - 1\}$ and $\ell \in \{0, \ldots, r(\lambda)\}$. Hybrid $H_{j,\ell}(1^\lambda)$ uses random

λ-bit values for $v_1, \ldots, v_j$ in the evaluation of F and sets v_λ to be random for the first ℓ unique queries. It follows that $H_{0,0}(1^\lambda)$ corresponds to using F as defined, and $H_{\lambda-1,r(\lambda)}(1^\lambda)$ corresponds to a random function.

The reduction R receives a challenge query y which is either sampled from $G(U_\lambda)$ or $U_{2\lambda}$. R samples $j \leftarrow \{0, \ldots, \lambda - 1\}$ and $\ell \leftarrow \{0, \ldots, r(\lambda) - 1\}$. For the first ℓ queries, it computes F using random values for $v_1, \ldots, v_{j+1}$ and computes pseudorandom values for the rest as defined by F For query $\ell + 1$, it uses the challenge y in place of v_{j+1}. For the remaining queries, it uses pseudorandom values defined by F for all $v_{j+1}, \ldots, v_\lambda$. After at most $r(\lambda)$ queries, A sends a bit $b^\star$, which R forwards to the PRG challenger.

Let $\delta = a - 1/2$. Let $p_{j,\ell}(\lambda)$ be the probability A outputs 1 when interacting with hybrid $H_{j,\ell}(1^\lambda)$. It follows that $p_{\lambda-1,r(\lambda)}(\lambda) - p_{0,0}(\lambda) = 2\delta$. Conditioned on a choosing $j \in \{0, \ldots, \lambda - 1\}$ and $\ell \in \{0, \ldots, r(\lambda) - 1\}$, R outputs 1 with probability $p_{j,\ell+1}(\lambda)/2 + (1 - p_{j,\ell}(\lambda))/2$. However, note that $p_{j,r(\lambda)}(\lambda) = p_{j+1,0}(\lambda)$ since $H_{j,r(\lambda)}(1^\lambda)$ is equivalent to $H_{j+1,0}(1^\lambda)$. It follows that R's overall success probability is

$$
\begin{aligned}
&\sum_{\substack{j \in \{0,\ldots,\lambda-1\} \\ \ell \in \{0,\ldots,r(\lambda)-1\}}} \frac{1}{\lambda \cdot r(\lambda)} \left(p_{j,\ell+1}(\lambda)/2 + (1 - p_{j,\ell}(\lambda))/2\right) \\
&= \sum_{j \in \{0,\ldots,\lambda-1\}} \frac{1}{\lambda \cdot r(\lambda)} \left(\frac{r(\lambda)}{2} + \frac{1}{2}(p_{j,r(\lambda)}(\lambda) - p_{j,0}(\lambda))\right) \\
&= \frac{1}{2} + \frac{1}{2 \cdot \lambda \cdot r(\lambda)} \cdot \left(p_{\lambda-1,r(\lambda)}(\lambda) - p_{0,0}(\lambda)\right) \\
&= \frac{1}{2} + \frac{\delta}{\lambda \cdot r(\lambda)},
\end{aligned}
$$

as required. □

B.3 IND-CPA Secure Encryption

An encryption scheme consists of three algorithms: a key generation algorithm KeyGen, an encryption algorithm Enc, and a decryption algorithm Dec. For a validly generated secret key, the encryption of any message m should correctly decrypt back to m, while also remaining "hidden" from any observer that does not have access to the secret key. There are a variety of ways to formalize this intuitive notion of security. In this paper, we focus on the notion of indistinguishability under chosen plaintext attack (IND-CPA), which is a multiple round game where the attacker only needs to distinguish encryptions of two messages *of its choice* after seeing any number of adaptively chosen encryptions.

Definition B.5 (Encryption Syntax and Correctness)**.** An encryption scheme is a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

1. $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$: A PPT algorithm that takes as input a security parameter 1^λ and outputs a secret key $\mathsf{sk} \in \{0,1\}^*$. We assume without loss of generality that sk always starts with 1^λ.

2. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$: A PPT algorithm that takes as input a secret key sk and a message $m \in \{0,1\}^\lambda$ and outputs a ciphertext $\mathsf{ct} \in \{0,1\}^*$.

3. $m' = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: A polynomial-time algorithm that takes as input a secret ket sk and a ciphertext ct and outputs a message m.

We say that an encryption scheme is correct if for all $\lambda \in \mathbb{N}$, $\mathsf{sk} \in \mathrm{Supp}\big(\mathsf{KeyGen}(1^\lambda)\big)$, $m \in \{0,1\}^\lambda$, $\mathsf{ct} \in \mathrm{Supp}(\mathsf{Enc}(\mathsf{sk}, m))$, it holds that $m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$.

Definition B.6 (r-Round IND-CPA Secure Encryption)**.** Let $r\colon \mathbb{N} \to \mathbb{N}$. The challenger C for the r-round IND-CPA secure encryption game for $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ interacts with an adversary A on common input 1^λ as follows.

- **Initialization:** C samples a secret key $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- **Pre-challenge phase:** A sends at most $r(\lambda)$ rounds of pre-challenge queries of the form $m_i \in \{0,1\}^\lambda$ for $i \in [r(\lambda)]$. C responds to each query with $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sk}, m_i)$.
- **Challenge phase:** A sends two challenge queries $m_0^\star$ and $m_1^\star$. C samples $b \leftarrow \{0,1\}$ and responds with $\mathsf{ct}^\star \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b^\star)$.
- **Post-challenge phase:** A sends at most $r(\lambda)$ rounds of post-challenge queries of the form $m_i' \in \{0,1\}^\lambda$ for $i \in [r(\lambda)]$. C responds to each query with $\mathsf{ct}_i' \leftarrow \mathsf{Enc}(\mathsf{sk}, m_i')$.
- **Output:** A sends a bit $b^\star$ and C outputs 1 iff $b = b^\star$.

From any PRF F, there is a classical construction of an IND-CPA secure encryption scheme (see e.g. [Gol09]). The security proof is single-shot, straightline black-box, so it also implies a universal reduction. The construction $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows.

- $\mathsf{KeyGen}(1^\lambda)$: Sample $K \leftarrow \{0,1\}^\lambda$ and output $\mathsf{sk} = (1^\lambda, K)$.
- $\mathsf{Enc}(\mathsf{sk}, m)$: Let $\mathsf{sk} = (1^\lambda, K)$. Sample $\rho \leftarrow \{0,1\}^\lambda$ and output $\mathsf{ct} = (\rho, F(K, \rho) \oplus m)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = (1^\lambda, K)$ and $\mathsf{ct} = (\rho, y)$. Output $m = F(K, \rho) \oplus y$.

Corollary B.3. *For any polynomial r, there exists a negligible function μ and an ϵ-universal reduction from the r-round security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ to the $2r+1$-round PRF security of F for $\epsilon(\lambda, a) = 1/2 + \delta/2 - \mu(\lambda)$, where $\delta = a - 1/2$.*

Proof sketch. We construct a single-shot straightline black-box reduction R by a hybrid argument, which implies a universal reduction by Theorem 4.1.

Let C_F be the $2r+1$-round PRF security game challenger, and let C be the r-round IND-CPA encryption challenger. We consider the following four hybrid challengers for the IND-CPA encryption game.

- $H_0(1^\lambda)$: Simulates C perfectly except always choose $b = 0$.
- $H_1(1^\lambda)$: Simulates C and always chooses $b = 0$. Additionally, samples a random function $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ and uses $f(\cdot)$ instead of $F(K, \cdot)$.
- $H_2(1^\lambda)$: Simulates C and always chooses $b = 1$. Also samples a random function $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ and uses $f(\cdot)$ instead of $F(K, \cdot)$.
- $H_3(1^\lambda)$: Simulates C perfectly except always choose $b = 1$.

Let A be an adversary with advantage $1/2 + \delta$ in C. We define $p_i(\lambda)$ to be the probability that A outputs $b' = 1$ in hybrid $H_i(1^\lambda)$ for all $i \in \{0, 1, 2, 3\}$. Note that, by assumption of A, it holds that $p_3(\lambda) - p_0(\lambda) = 2\delta$.

The reduction R acts as an adversary in the game C_F while interacting with A as a challenger in an r-round IND-CPA encryption game. R simulates the real challenger C, except that any time it needs to compute $F(K, x)$ on an input $x \in \{0,1\}^\lambda$, it queries the PRF challenger C_F on input x to get the corresponding value. Note that this requires R to make at most $2r + 1$ queries. If the challenger C_F samples the challenge bit $b = 1$, the output will be from a random function, and otherwise if $b = 0$, it will be from a pseudorandom function. Let $b' \leftarrow \{0, 1\}$ be the bit chosen by R during the challenger phase. In the output phase, R receives a bit $b^\star$ from A. If $b' = 1$, R sends $1 - b^\star$ to C_F as its output. If $b' = 0$, it will send $b^\star$ to C_F.

We break down the success of R conditioned on the value of b, b' in the following cases:

- $b = 0$ and $b' = 0$ corresponds to $H_0(1^\lambda)$, so R wins with probability $(1 - p_0(\lambda))$
- $b = 1$ and $b' = 0$ corresponds to $H_1(1^\lambda)$, so R wins with probability $p_1(\lambda)$
- $b = 1$ and $b' = 1$ corresponds to $H_2(1^\lambda)$, so R wins with probability $(1 - p_2(\lambda))$
- $b = 0$ and $b' = 1$ corresponds to $H_3(1^\lambda)$, so R wins with probability $p_3(\lambda)$.

So overall, the reduction R wins with probability

$$\begin{aligned}
&(1/4) \cdot (1 - p_0(\lambda)) + (1/4) \cdot p_1(\lambda) + (1/4) \cdot (1 - p_2(\lambda)) + (1/4) \cdot p_3(\lambda) \\
&= 1/2 + (1/4) \cdot (p_3(\lambda) - p_0(\lambda) + p_2(\lambda) - p_1(\lambda)) \\
&= 1/2 + \delta/2 + (p_2(\lambda) - p_1(\lambda))/4.
\end{aligned}$$

It remains to show that $(p_2(\lambda) - p_1(\lambda))/4$ is negligible. Unless the reduction R samples the same value of $\rho \in \{0,1\}^\lambda$ over the $2r + 1$ many encryption queries, the ITMs $H_2(1^\lambda)$ and $H_1(1^\lambda)$ are identically distributed since that implies output of the random function is uniform for each query. Thus, $f(\rho)$ perfectly masks the message m and reveals statistically no information. It follows that for any, even ubounded, adversary A, $p_2(\lambda) - p_1(\lambda) \leq (2r + 1)^2/2^\lambda$. Therefore, $(p_2(\lambda) - p_1(\lambda))/4$ is negligible, and the corollary follows. □

B.4 Commitments

A commitment scheme consists of two parties, a sender and a receiver. Commitments allow a sender to "commit" itself to a bit at some point in time that will be revealed to the receiver at a later point. The receiver should not be able to guess the committed bit before the sender reveals, this is known has *hiding*, and the sender should not be able to reveal two different values, this is known as *binding*. Commitments come in two flavors: binding commitments, where the binding property is statistical and the hiding property is computational, and hiding commitments, where the hiding property is statistical and the binding property is computational. We will only consider binding commitments for the sake of this paper, which we formalize below.

We restrict our attention to 2-message, binding, bit commitment schemes for simplicity. We can extend this to commitments for longer messages, or a set of messages, by parallel repetition of a single scheme. The syntax and security games extend to this more general case by simply increasing the length of the messages.

Definition B.7 (Commitment Syntax)**.** A 2-message bit commitment scheme consists of a pair of PPT algorithm $(\mathsf{Gen}, \mathsf{Commit})$ with the following syntax:

- $\sigma \leftarrow \mathsf{Gen}(1^\lambda)$: A PPT algorithm that takes as input a security parameter 1^λ and outputs a string $\sigma \in \{0,1\}^*$.
- $\mathsf{com} = \mathsf{Commit}(1^\lambda, m, \sigma; r)$: A PPT algorithm that uses randomness $r \in \{0,1\}^\lambda$, takes as input a security parameter 1^λ, message $m \in \{0,1\}$, and first message $\sigma \in \{0,1\}^*$ from the receiver, and outputs a commitment value $\mathsf{com} \in \{0,1\}^*$. The randomness r is known as the opening for the commitment.

In the above definition, if the first message output by Gen is always the empty string, we say that the scheme is a 1-message commitment scheme. We next formalize the binding and hiding security games for a commitment scheme.

Definition B.8 (Binding Security Game)**.** The challenger C for the binding security game of $(\mathsf{Gen}, \mathsf{Commit})$ interacts with an adversary A on common input 1^λ as follows. C samples $\sigma \leftarrow \mathsf{Gen}(1^\lambda)$ and sends σ to A. A responds with a commitment value com, messages m_0 and m_1, openings r_0 and r_1. C outputs 1 iff $\mathsf{com} = \mathsf{Commit}(1^\lambda, m_0, \sigma; r_0)$ and $\mathsf{com} = \mathsf{Commit}(1^\lambda, m_1, \sigma; r_1)$.

Definition B.9 (Hiding Security Game)**.** The challenger C for the hiding security game for $(\mathsf{Gen}, \mathsf{Commit})$ interacts with an adversary A on common input 1^λ as follows. A outputs two messages m_0 and m_1. C samples $b \leftarrow \{0,1\}$, $\sigma \leftarrow \mathsf{Gen}(1^\lambda)$, and $\mathsf{com} \leftarrow \mathsf{Commit}(1^\lambda, m_b, \sigma)$. C sends com and σ to A. A responds with a bit $b^\star$, and C outputs 1 iff $b = b^\star$.

Naor [Nao91] gives a construction of a bit commitment scheme from any 3λ-bit stretch PRG G. The security proof of this construction is a single-shot, straightline black-box reduction, so it immediately gives a universal reduction. The construction $(\mathsf{Gen}, \mathsf{Commit})$ is defined as follows.

- $\mathsf{Gen}(1^\lambda)$: Output $\sigma \leftarrow \{0,1\}^{3\lambda}$.
- $\mathsf{Commit}(1^\lambda, m, \sigma; r)$: If $m = 0$, output $G(r)$. If $m = 1$, output $\sigma \oplus G(r)$.

This commitment scheme can be run in parallel to get a commitment scheme for longer messages. This gives a security loss that depends on the length of the message.

We note that binding for this construction is unconditional, so we focus instead on the computational hiding property.

Corollary B.4. *Let G be a 3λ-bit stretch PRG. There exists an ϵ-universal reduction from the hiding of $(\mathsf{Gen}, \mathsf{Commit})$ to the PRG security of G for $\epsilon(\lambda, a) = 1/2 + \delta/2$, where $\delta = a - 1/2$.*

Proof sketch. We provide a single-shot, straightline, black-box reduction R using a hybrid argument, which implies a universal reduction by Theorem 4.1.

Let C be the challenger for the hiding security game and C' be the challenger for the PRG security game. We consider the four hybrid challengers for the hiding security game.

- $H_0(1^\lambda)$: Simulates C perfectly except always choose $b = 0$.
- $H_1(1^\lambda)$: Simulates C and always chooses $b = 0$. Instead of using $G(r)$ in the construction, it uses $y \leftarrow U_{3\lambda}$ instead.

- $H_2(1^\lambda)$: Simulates C but chooses $b = 1$. Instead of using $G(r)$ in the construction, it uses $y \leftarrow U_{3\lambda}$.
- $H_3(1^\lambda)$: Simulates C perfectly except always choose $b = 1$.

The reduction R interacts with a challenger C' for the PRG security game using an attacker A that has advantage $1/2 + \delta$ in the hiding security game. We let $p_i(\lambda)$ be the probability that A outputs 1 when interacting in hybrid $H_i(1^\lambda)$. The reduction R is defined as follows.

C' samples a bit $b \leftarrow \{0,1\}$ and sends R a challenge y which is $G(U_\lambda)$ if $b = 0$ or $U_{3\lambda}$ if $b = 1$. R also receives two messages m_0 and m_1 from the hiding adversary A. R flips a bit $b' \leftarrow \{0,1\}$, samples $\sigma \leftarrow \mathsf{Gen}(1^\lambda)$, and sends A the values σ and $\mathsf{com} \leftarrow \mathsf{Commit}(1^\lambda, m_{b'}, \sigma)$. R receives a response $b^\star$ from A. If $b' = 1$, R sends $1 - b^\star$ to C'. If $b' = 0$, R sends $b^\star$ to C'.

If $b = 0, b' = 0$, the reduction R corresponds to $H_0(1^\lambda)$, so R wins with probability $(1 - p_0(\lambda))$. If $b = 1, b' = 0$, R corresponds to $H_1(1^\lambda)$, so R wins with probability $p_1(\lambda)$. If $b = 1, b' = 1$, R corresponds to $H_2(1^\lambda)$, so R wins with probability $(1 - p_2(1^\lambda))$. If $b = 0, b' = 1$, R corresponds to $H_3(1^\lambda)$, so R wins with probability $p_3(\lambda)$. Putting these observations together, we conclude that R wins with overall probability

$$\begin{aligned}
&(1/4) \cdot (1 - p_0(\lambda)) + (1/4) \cdot p_1(\lambda) + (1/4) \cdot (1 - p_2(\lambda)) + (1/4) \cdot p_3(\lambda) \\
&= 1/2 + (1/4) \cdot (p_3(\lambda) - p_0(\lambda) + p_2(\lambda) - p_1(\lambda)) \\
&= 1/2 + \delta/2 + (p_2(\lambda) - p_1(\lambda))/4.
\end{aligned}$$

It remains to show that $p_2(\lambda) - p_1(\lambda) = 0$. This is because $H_1(1^\lambda)$ and $H_2(1^\lambda)$ are identically distributed. Regardless of if $b' = 0$ or $b' = 1$, R outputs a uniformly random string in $\{0,1\}^{3\lambda}$. Thus, R's advantage is simply $1/2 + \delta/2$, as required. □

B.5 One-Time Signatures

Signature schemes consist of a signer and a verifier. Such a scheme allows a signer send a message such that the verifier can authenticate that the signer indeed generated the message. we focus on signature schemes for fixed length messages, where we only require security for a single signed message. These are known as one-time signatures. We start by defining the syntax for a signature scheme in general.

Definition B.10 (Signature Syntax and Correctness)**.** A signature scheme consists of a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ with the following syntax:

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$: A PPT algorithm that takes as input a security parameter 1^λ and outputs a signing key $\mathsf{sk} \in \{0,1\}^*$ and a verification key $\mathsf{vk} \in \{0,1\}^*$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$: A PPT algorithm that takes as input a signing key $\mathsf{sk} \in \{0,1\}^*$ and a message $m \in \{0,1\}^\lambda$ and outputs a signature $\sigma \in \{0,1\}^*$.
- $b = \mathsf{Ver}(\mathsf{vk}, m, \sigma)$: A deterministic polynomial time algorithm that takes as input a verification key $\mathsf{vk} \in \{0,1\}^*$, a message $m \in \{0,1\}^\lambda$, and a signature $\sigma \in \{0,1\}^*$ and outputs a bit b indicating whether to accept or reject the signature.

A signature scheme is correct if for all $\lambda \in \mathbb{N}$, $m \in \{0,1\}^\lambda$, $(\mathsf{sk}, \mathsf{vk}) \in \mathrm{Supp}\big(\mathsf{Gen}(1^\lambda)\big)$, and $\sigma \in \mathrm{Supp}(\mathsf{Sign}(\mathsf{sk}, m))$, it holds that $\mathsf{Ver}(\mathsf{vk}, m, \sigma) = 1$.

Definition B.11 (One-Time Signature Security)**.** The challenger C for the one-time signature security game for $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ interacts with an adversary A on common input 1^λ as follows. C samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends vk to A. A sends a single message $m \in \{0,1\}^\lambda$ to C, and C responds with $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$. A then sends $m^\star, \sigma^\star$ to C, who accepts iff $m^\star \neq m$ and $\mathsf{Ver}(\mathsf{vk}, m^\star, \sigma^\star) = 1$.

We recall Lamport's one-time signature construction based on any one-way function f [Lam79]. As its security is based on a single-shot, straightline black-box reduction, it implies a universal reduction.

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$: For $i \in [\lambda]$ and $j \in \{0,1\}$, sample $x_{i,j} \leftarrow \{0,1\}^\lambda$ and let $y_{i,j} = f(x_{i,j})$. Output sk, vk where $\mathsf{sk} = \{x_{i,j}\}_{i\in[\lambda], j\in\{0,1\}}$ and $\mathsf{vk} = \{y_{i,j}\}_{i\in[\lambda], j\in\{0,1\}}$.
- $\sigma \leftarrow \sigma(\mathsf{sk}, m)$: Let $\sigma_i = x_{i,m_i}$ and output $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$.
- $b = \mathsf{Ver}(\mathsf{vk}, m, \sigma)$: Output 1 iff $f(\sigma_i) = y_{i,m_i}$ for all $i \in [\lambda]$.

We outline the proof of one-time security of this construction below to show how it implies a universal reduction to one-way function security.

Corollary B.5. *There exists an ϵ-universal reduction from the one-time security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ to the one-way function security of f for $\epsilon(\lambda, a) = a/(2\lambda)$.*

Proof sketch. We provide a single-shot, straightline, black-box reduction R, which in turn implies a universal reduction by Theorem 4.1.

Let C be the challenger in the one-time security game and C' be the challenger for the one-way function security game. Fix $\lambda \in \mathbb{N}$. The reduction R interacts with an instance of C' making use of an adversary A that wins C with probability a. R is defined as follows.

R receives a challenge $y \leftarrow f(U_\lambda)$ from C'. It then samples a random $i \leftarrow [\lambda]$ and $j \leftarrow \{0,1\}$. For all $(i', j') \neq (i, j) \in [\lambda] \times \{0,1\}$, R samples $x_{i',j'} \leftarrow \{0,1\}^\lambda$ and computes $y_{i',j'} = f(x_{i',j'})$. R sets $y_{i,j} = y$ from the challenger. R sends the verification key $\{y_{i',j'}\}_{i'\in[\lambda], j'\in\{0,1\}}$ to A and receives a message $m \in \{0,1\}^\lambda$ to sign. If $m_i = j$, R aborts. Otherwise, R sets $\sigma_{i'} = x_{i',m_{i'}}$ for all $i' \in [\lambda]$ and sends $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$ to A. A outputs a message $m^\star, \sigma^\star$. If $m^\star_i = j$, let $x = \sigma^\star_i$. If $f(x) = y$, R sends x to the challenger C'.

If A wins, it must be the case that $m \neq m^\star$ and $f(\sigma^\star_i) = y$. As i and j are chosen uniformly and independently from $[\lambda]$ and $\{0,1\}$, respectively, the probability that m and $m^\star$ differ in the ith bit, $m_i \neq j$, and A wins is at least $a/(2\lambda)$, over a random i and j. Otherwise, A must win with probability less than a in total. Therefore, R has success probability $a/(2\lambda)$, as required. □