Immunizing Backdoored PRGs

Marshall Ball[*], Yevgeniy Dodis[†], and Eli Goldin[‡]

New York University

Abstract

A backdoored Pseudorandom Generator (PRG) is a PRG which looks pseudorandom to the outside world, but a saboteur can break PRG security by planting a backdoor into a seemingly honest choice of *public parameters*, pk, for the system. Backdoored PRGs became increasingly important due to revelations about NIST's backdoored Dual EC PRG, and later results about its practical exploitability.

Motivated by this, at Eurocrypt'15 Dodis et al. [21] initiated the question of *immunizing* backdoored PRGs. A k-immunization scheme repeatedly applies a post-processing function to the output of k backdoored PRGs, to render any (unknown) backdoors provably useless. For $k = 1$, [21] showed that no deterministic immunization is possible, but then constructed "seeded" 1-immunizer either in the random oracle model, or under strong non-falsifiable assumptions. As our first result, we show that no seeded 1-immunization scheme can be black-box reduced to any efficiently falsifiable assumption.

This motivates studying k-immunizers for $k \geq 2$, which have an additional advantage of being deterministic (i.e., "seedless"). Indeed, prior work at CCS'17 [37] and CRYPTO'18 [7] gave supporting evidence that simple k-immunizers might exist, albeit in slightly different settings. Unfortunately, we show that simple standard model proposals of [37, 7] (including the XOR function [7]) provably do not work in our setting. On a positive, we confirm the intuition of [37] that a (seedless) random oracle is a provably secure 2-immunizer. On a negative, no (seedless) 2-immunization scheme can be black-box reduced to any efficiently falsifiable assumption, at least for a large class of natural 2-immunizers which includes all "cryptographic hash functions."

In summary, our results show that k-immunizers occupy a peculiar place in the cryptographic world. While they likely exist, and can be made practical and efficient, it is unlikely one can reduce their security to a "clean" standard-model assumption.

1 Introduction

Pseudorandom number generators (PRGs) expand a short, uniform bit string s (the "seed") to a larger sequence of pseudorandom bits X. Beyond their status as a fundamental primitive in cryptography, they are used widely in practical random number generators, including those in all major operating systems. Unsurprisingly, PRGs have been target of many attacks over the years. In this work we focus on a specific, yet prominent, type of PRG attack which arises by planting a *backdoor* inside the PRG. This type of attack goes far back to 1983, when Vazirani and

[*]Electronic address: marshall@cs.columbia.edu. Supported in part by the Simons Foundation.

[†]Electronic address: dodis@cs.columbia.edu. Research Supported by NSF grant CNS-2055578, and gifts from JP Morgan, Protocol Labs and Algorand Foundation.

[‡]Electronic address: eg3293@nyu.edu. Partially supported by a National Science Foundation Graduate Research Fellowship.

Vazirani [42, 43] introduced the notion of "trapdoored PRGs" and showed the Blum-Blum-Shub PRG is one such example [12]. Their purpose was not for sabotaging systems, however, but instead they used the property constructively in a higher level protocol.

NIST Dual EC PRG. Perhaps the most infamous demonstration of the potential for sabotage is the backdoored NIST Dual EC PRG [1]. Oversimplifying this example for the sake of presentation (see [39, 16, 21] for the "real-world" description), the attack works as follows. The (simplified) PRG is parameterized by two elliptic curve points; call them P and Q. These points are supposed to be selected at random and independent from each other, forming the PRG public parameter $pk = (P, Q)$ which can be reused by multiple PRG instances. Each new PRG instance then selects a random initial seed s, and can expand into random-looking elliptic curve points $X = sP$ and $Y = sQ$. Ignoring the details of mapping elliptic curve points into bit-strings,[1] as well as subsequent iterations of this process, one can conclude that the points (X, Y) are pseudorandom conditioned on $pk = (P, Q)$. In fact, this is *provably so* under to widely believed Decisional Diffie-Hellman (DDH) assumption.

Yet, imagine that the entity selecting points P and Q chooses the second point Q as $Q = dP$ for a random multiple ("discrete log") d, and secretly keeps this multiple as its backdoor $sk = d$. Notice, the resulting public parameter distribution $pk = (P, Q)$ is *identical* to the supposed "honest" distribution, when Q was selected independently from P. Thus, the outside world cannot detect any cheating in this step, and could be swayed to use the PRG due to its provable security under the DDH assumption. Yet, the knowledge of d can easily allow the attacker to distinguish the output (X, Y) from random; or, worse, predict Y from X, by noticing that

$$Y = sQ = s(dP) = d(sP) = dX$$

While we considerably simplified various low level details of the Dual EC PRG, the works of [39, 16] showed that the above attack idea can be extended to attacking the actual NIST PRG. Moreover, the famous "Juniper Dual EC incident" (see [15] and references therein) showed that this vulnerability was likely used for years in a real setting of Juniper Networks VPN system!

Backdoored PRGs. Motivated by these real-world considerations, the work of Dodis et al. [21] initiated a systematic study of so called *backdoored PRGs*, abstracting and generalizing the Dual EC PRG example from above. A backdoored PRG (K, G) is specified by a (unknown to the public) key generation algorithm K which outputs public parameters pk, and a hidden backdoor sk. The "actual PRG" G takes pk and a current PRG state s as input, and generates the next block of output bits R and the updated (internal) state s. The initial seed/state $s = s_0$ is assumed to be chosen at random and not controlled/sabotaged by the attacker. We call this modeling *honest initialization*, emphasizing that the Dual EC PRG attack was possible even under such assumption. The PRG can then be iterated any number of times q, producing successive outputs (R_i) and corresponding internal states (s_i). The basic constraint on the saboteur is that the joint output $X = (R_1, \ldots, R_q)$ should be indistinguishable from uniform given only the public parameters pk (but not the secret backdoor sk). We call this constraint *public security*.

Unfortunately, the dual EC PRG example shows that public security — even when accompanied by a "security proof" — does not make the backdoor PRG secure against the *saboteur, who also knows sk*. In fact, [21] showed that the necessary and sufficient assumption for building effective backdoor PRGs (secure to public but broken using sk) is the existence of any public-key encryption

[1] And instead thinking of PRG as outputting pseudorandom elliptic curve points.

scheme with pseudorandom ciphertexts.

1.1 Our Questions: Immunization Countermeasures

While the question of designing backdoored PRGs is fascinating, in this work we are interested in various countermeasures against backdoor PRGs, a topic of interest given the reduced trust in PRGs engendered by the possibility of backdooring. Obviously, the best countermeasure would be to use only trusted PRGs, if this is feasible. Alternatively, one could still agree to use a given backdoor PRG, but attempt to overwrite its public parameters pk. For example, this latter approach is advocated (and formally proven secure) in [35, 5]. Unfortunately, these techniques cannot be applied in many situations. For example, existing proprietary software or hardware modules may not be easily changed, or PRG choices may be mandated by standards, as in the case of FIPS. Additionally, the user might not have not have direct control over the implementation itself (for example, if it is implemented in hardware or the kernel), or might not have capability or expertise to properly overwrite (potentially hidden or hardwired) value of pk. Fortunately, there is another approach which is much less intrusive, and seems to be applicable to virtually any setting: to efficiently *post-process the output* of a PRG in an online manner in order to prevent exploitation of the backdoor. We call such a post-processing strategy an *immunizer*.[2]

The question of building such immunizers was formally introduced and studied by Dodis et al. [21]. For example, the most natural such immunizer would simply apply a cryptographic hash function C, such as SHA-256 (or SHA-3), to the current output R_i of the PRG, only providing the saboteur with value $Z_i = C(R_i)$ instead of R_i itself. The hope being that hashing the output of a PRG will provide security even against the suspected backdoor sk.[3] Unfortunately, [21] showed that this natural immunizer does not work in general, even if C is modeled as a Random Oracle (RO)! Moreover, this result easily extends to any deterministic immunizer C (e.g., bit truncation, etc).

Instead, the solution proposed by [21] considers a weaker model of probabilistic/seeded immunizers, where it is assumed that some additional, random-but-public parameter can be chosen after the attacker finalized design of the backdoor PRG (K, G), and published the public parameters pk. While [21] provide some positive results for such *seeded immunizers*, these results were either in the random oracle model, or based on the existence of so called universal computational extractors (UCEs) [8]. Thus, we ask the question:

Question 1. *Can one built a seeded backdoor PRG immunizer in the standard model, under an efficiently falsifiable*[4] *assumption?*

As our first result, we use the elegant black-box separation technique of Wichs [44] to give a negative answer to this question (see Section 5.4).

Theorem 1.1. *If there is a black-box reduction showing security of seeded immunizer C from the security of some cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

[2]Note that the immunizer only processes pseudorandom outputs and does not have access to the internal state (which is not necessarily available to a user). Indeed, if one has access to a random initial state, there is a trivial "immunizer" that ignores the given backdoor PRG, and instead uses the random state to bootstrap a different (non-backdoored) PRG.

[3]This assumption presumes that such C itself is not backdoored.

[4]Recall that, loosely speaking, an assumption is efficiently falsifiable if the falseness of the assumption can be verified (efficiently), given an appropriate witness.

Moreover, the availability and trust issues in generating and agreeing on the public seed required for the immunization make this solution undesirable or inapplicable for many settings. Thus, we ask the question if *deterministic* immunizers could exist in another meaningful model, despite the impossibility result of [21] mentioned above. And, as a secondary question, if they can be based on efficiently falsifiable assumptions.

2-Immunizers to Rescue? We notice that the impossibility result of [21] implicitly (but *critically*) assumes that only a *single* honestly-initialized backdoor PRG is being immunized. Namely, the immunizer C is applied to the output(s) R_i of a single backdoor PRG (K, G). Instead, we notice that many PRGs allow to explicitly initialize *multiple independent copies*. For example, a natural idea would be to initialize two (random and independent) initial states s and s' of the PRG, run these PRGs in parallel, but instead of directly outputting these outputs R_i and R'_i, respectively, the ("seedless") immunizer C will output the value $Z_i = C(R_i, R'_i)$ to the attacker.[5] We call such post-processing procedures 2-*immunizers*.[6] More generally, one can consider k-immunizers for $k \geq 2$, but setting $k = 2$ is obviously the most preferable in practice. As before, our hope would be that the final outputs $(Z_1, \ldots, Z_q)$ will be pseudorandom even conditioned on the (unknown) backdoor sk, and even if the key generation algorithm K could depend on the choice of our 2-immunizer C. This is the main question we study in this work:

Question 2 (Main Question)**.** *Can one construct a provably secure* 2*-immunizer* C *against all efficient backdoored PRGs* (K, G)*?*

We note that several natural candidates for such 2-immunizers include XOR, inner product, or a cryptographic hash function C.

A note on immunizers from computational assumptions One may wonder whether it is worth considering immunizers whose security depends on a computational assumption. After all, if the computational assumption is sufficiently strong to imply that pseudorandom generators exist (as most assumptions are), then why would we not just use the corresponding PRG? However, we think that building a immunizer in this setting is still interesting for two reasons. First, if we can show that a immunizer exists in this regime, then this gives evidence that an information-theoretic style immunizer also exists. Second, there are some scenarios where one has access to PRG outputs but no access to true randomness (for example if the kernel does not give direct access to its random number generator). In this setting, we can use a computational immunizer to recover full security.

1.2 Related Immunization Settings

Before describing our results, it might be helpful to look at the two conceptually similar settings considered by Bauer at al. [7, 20] and Russell et al. [37].

Detour 1: Backdoored Random Oracles. In this model [7], one assumes the existence of a truly random oracle G. However, the fact that G might have been "backdoored" is modeled by providing the attacker with the following *leakage oracle* any polynomial number of times: given any

[5] Note that again if the post-processing is not sufficiently "simple" (here this means statelessly processing outputs in an online manner), one can trivially bootstrap "honest" public parameters from many fresh PRG invocations.

[6] Drawing inspiration from 2-source extractors [17] to similarly overcome the impossibility of deterministic extraction from a single weak source of randomness.

(potentially inefficient) function g, the attacker can learn the output of g applied to the entire truth-table of G. For example, one can trivially break the PRG security of a length-expanding random oracle $R = G(s)$, by simply asking the leakage oracle $g_R(G)$ whether there is a shorter-than-R seed s s.t. $G(s) = R$.

With this modeling, [7] asked (among other things) whether one can build 2-immunizers for two independent BROs F and G. For example, in case of pseudorandomness, they explicitly asked if $H(s) = F(s) \oplus G(s)$ is pseudo-random (for random seed s), even if the distinguisher can have polynomial number of leakage oracle calls to F and G separately (but not jointly). Somewhat surprisingly, they reduce this question to a plausible conjecture regarding communication complexity of the classical set-intersection problem (see [14] for a survey of this problem). Thus, despite not settling this question unconditionally, the results of [7] suggest that XOR might actually work for the case of PRGs.

In addition, [38] studies the question of k-immunizers in the related setting of "subverted" random oracles (where the subverted oracle differs from the true one on a small number of inputs). There, a simple yet slightly more complicated "xor-then-hash" framework is shown to provide a good immunizer.

Detour 2: Kleptographic Setting. While the study of kleptography goes back to the seminal works of Young and Yung [45, 46, 47] (and many others), let us consider a more recent variant of [37]. This model is quite general, attempting to formalize the ability of the public to test if a given black-box implementation is done according to some ideal specification. As a special case, this could in particular cover the problem of public parameter subversion of PRGs, where the PRG designer kept some secret information sk, instead of simply choosing pk at random.

We will comment on the subtleties "kleptographic PRGs" vs "backdoored PRGs" a bit later, but remark that [37] claimed very simple k-immunizers in their setting. Specifically they showed that for one-shot PRGs (where there is no internal state for deriving arbitrarily many pseudorandom bits) in the kleptographic setting, random oracle C is a good 2-immunizer, while for $k \gg 2$, one can even have very simple k-immunizers in the standard model. For example, have each of k PRGs shrink its output to a single bit, and then concatenate these bits together. Again this suggests that something might work for the more general case of (stateful) PRGs.

1.3 Our Results for 2-Immunizers

As we see, in both of these related settings it turns out that simple k-immunizers exist, including XOR and random oracle for $k = 2$. Can these positive results be extended to the backdoored PRG setting?

XOR is Insecure. First we start with the simple XOR 2-immunizer $C(x, y) = x \oplus y$, which is probably the simplest and most natural scheme to consider. Moreover, as we mentioned, the PRG results of [7] for BROs give some supporting evidence that this 2-immunizer might be secure in the setting of backdoor PRGs. Unfortunately, we show that this is not the case.[7] Intuitively, the BRO modeling assumes that both generators F and G are modeled as true random oracles with bounded leakage, which means that both of them have a lot of entropy hidden from the attacker. In contrast, the backdoor PRG model of [21] (and this work) allows the attacker to build F and G

[7] Under a widely believed cryptographic assumption mentioned shortly.

which are extremely far from having any non-trivial amount of entropy to the attacker who knows the backdoor sk.

Indeed, our counter-example for the XOR immunizer comes from a more general observation, which rules out all 2-immunizers C for which one can build a public key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ which has pseudorandom ciphertexts, and is what we call *C-homomorphic*. Oversimplifying for the sake of presentation (see Definition 3.5), we need an encryption scheme where the message m — independently encrypted twice under the same public key pk with corresponding ciphertexts x and y — can still be recovered using the secrete key sk and "C-combined" ciphertext $z = C(x, y)$. If such a scheme exists, the backdoor PRG can simply output independent encryptions of a fixed message (say, 0) as its pseudorandom bits. The C-homomorphic property then ensures that the attacker can still figure that 0 was encrypted after seeing the combined ciphertext $z = C(x, y)$, where x and y are now (individually pseudorandom, and hence secure to public) encryptions of 0. Moreover, we build a simple "XOR-homomorphic" public key encryption under a variant of the LPN assumption due to Alekhnovich [3]. Thus, under this assumption we conclude that XOR is not a secure 2-immunizer.

Theorem 1.2. *Assuming the Alekhnovich assumption (listed in Proposition 3.7) holds, XOR is not a secure 2-immunizer.*

Inadequacy of Kleptographic Setting for PRGs. Our second observation is that the kleptographic setting considered by [37] — which extremely elegant and useful for many other cryptographic primitives (and additionally considers the dimension of corrupted implementations, which we do not consider) – does not adequately model the practical problem of backdoored PRGs. In essence, the subverted PRG modeling of [37, 36] yields meaningful results in the stateless (one-time output production) setting, but does not extend to the practically relevant stateful setting. It is worth noting that while [37] informally claim (see Remark 3.2 in [36]) a trivial composition theorem to move from stateless to the (practically relevant) stateful setting, that result happens to be vacuous.[8] In particular, the "ideal specification" of stateful PRGs (implicitly assumed by the authors in their proofs) requires that stateful PRG would produce fresh and unrelated outputs, even after rewinding the PRG state to some prior state. However, PRGs are deterministic after the initial seed is chosen. As such, even the most secure and "stego-free" implementation will never pass such rewinding test, as future outputs are predetermined once and for all. Stated differently, the "ideal specification" of stateful PRG implicitly assumed by [37, 36] in Remark 3.2 is too strong, and no construction can meet it.[9]

To see this modeling inadequacy directly, recall that one of the standard model k-immunizers from [37, 36] simply concatenates the first bit of each PRG's output. For a stateless (one-time) PRG case, this is secure for trivial (and practically useless) reasons: each PRG bit should be statistically random, or the "public" (called the "watchdog" by the authors) will easily catch it. But now let us look at the stateful extension, — which could be potentially useful if it was secure, — and apply it to the the following Dual-EC variant. On a given initial state s, in round i the variant will output the ith bit of Dual-EC initialized with s. Syntactically, this is the same (very dangerous)

[8] In general, we conjecture no such composition result is true under proper modeling of backdoor PRGs, such as the one in this work. For example, 2-immunization for stateless PRGs can be effectively instantiated with a sufficiently strong 2-source extractor. In contrast, our negative result (mentioned later in the Introduction) rules out such extractors as sufficient for stateful PRGs.

[9] Note however, that their modeling does capture *pseudorandom number generators (PRNGs)* which accumulate entropy albeit in a setting where one has rewinding access and the entropy sources are not too adversarial.

backdoor PRG we would like to defend against, although made artificially less efficient. Yet, when the "concatenation" k-immunizer above is applied to this (stateful) variant, the attacker still learns full outputs of each of the k PRG copies, and can just do the standard attack on Dual-EC separately on each copy. This means that this k-immunizer is blatantly insecure in our setting, for any value of k.

Random Oracle is Secure. Despite the inability to generically import the positive results of [37, 36] to our setting, we can still ask if the random oracle 2-immunizer result claimed by [37, 36] is actually true for backdoored PRGs. Fortunately, we show that this is indeed the case, by giving a direct security proof.[10] In fact, it works even is the so called *auxiliary-input ROM* (AI-ROM) defined by Unruh [41] and recently studied by [22, 18]. In this model we allow the saboteur to prepare the backdoor sk and public parameters pk after *unbounded preprocessing* of the Random Oracle C. The only constraint of the resulting backdoored PRG G is that it has to be secure to the public in the standard ROM (since the public might not have enough resources to run the expensive preprocessing stage). Still, when being fed with outputs $z_i = C(x_i, y_i)$, the saboteur cannot distinguish them from random even given its polynomial-sized backdoor sk (which also models whatever auxiliary information about RO C the attacker computed), and additional polynomial number of queries to C.

Despite appearing rather expected, the proof of this result is quite subtle. It uses the fact that each independently initialized PRG instances F and G are unlikely to ever query the random oracle on any of the outputs produced by the other instance (i.e., F on $C(\cdot, y_i)$ and G on $C(x_i, \cdot)$), because we show that this will contradict the assumed PRG security of F and G from the public.

Theorem 1.3. *$C(X, Y) = RO(X||Y)$ is a secure 2-immunizer in the AI-ROM.*

Back-box Separation From Efficiently Falsifiable Assumptions. Finally, we consider the question of building a secure 2-immunizer in the standard model. In this setting, we again use the black-box separation technique of Wichs [44] to show the following negative result. No function $C(x, y)$, which is *highly dependent on both inputs x ad y*, can be proven as a secure 2-immunizer for backdoor PRGs, via a black-box reduction to any efficiently falsifiable assumption.

The formal definition of "highly dependent" is given in Definition **??**, but intuitively states that there are few "influential" inputs x^* (resp., y^*) which fix the output of C to a constant, irrespective of the other input. We notice that most natural functions are clearly highly dependent on both inputs. *This includes XOR, the inner product function, and any cryptographic hash function heuristically replacing a random oracle, such as SHA-256 or SHA-3.*

The latter category is unfortunate, though. While our main positive result gave plausible evidence that cryptographic hash functions are likely secure as 2-immunizers, our negative result shows that there is no efficiently falsifiable assumption in the standard model under which we can formally show security of any such 2-immunizer C.

Theorem 1.4. *Let C be a 2-immunizer which is highly dependent on both inputs. If there is a black-box reduction showing that C is secure from the security of some cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

[10] In particular, the key piece of our proof that was missing in [37, 36], is contained in Lemma 4.13 of our paper. The important observation (adapted from the seeded 1-immunizers proof in[21]) is that the random oracle outputs reveal negligible information about its inputs, and so every PRG round can inductively be treated as the first round.

WEAK 2-IMMUNIZERS. Given our main positive result is proven in the random oracle model, we also consider another meaningful type of immunizer which we call *weak* 2-*immunizer*, in hope that it might be easier to instantiate in the standard model. (For contrast, we will call the stronger immunizer concept considered so far as *strong* 2-*immunizer*.) Recall, in the strong setting the immunizer C was applied to two independently initialized copies of the *same* backdoor PRG (K, G). In particular, both copies shared the same public parameters pk. In contrast, in the weak setting, — in addition to independent seed initialization above, — we assume the backdoor PRGs were designed by two *independent key generation processes* K *and* K', producing independent key pairs (pk, sk) and (pk', sk'). For example, this could model the fact that competing PRGs were designed by two different standards bodies (say, US and China). Of course, at the end we will allow the two saboteurs to "join forces" and try to use both sk and sk' when breaking the combined outputs $Z_i = C(R_i, R'_i)$. Curiously, it is not immediately obvious that a strong 2-immunizer is also a weak one, but we show that this is indeed the case, modulo a small security loss. In particular, this implies that our positive result in the random oracle model also gives a weak 2-immunizer.

Of course, the interesting question is whether the relaxation to the weak setting makes it easier to have standard model instantiations. Unfortunately, *we show that this does not appear to be the case*, by extending most of our impossibility/separation results to the weak setting (as can be seen in their formal statements). The only exception is the explicit counter-example to the insecurity of XOR as a weak 2-immunizer, which we leave open (but conjecture to be true). As partial evidence, we show that the pairing operation (which looks similar to XOR) is not a weak 2-immunizer under a widely believed SXDH assumption in pairing based groups [6, 4].

Theorem 1.5. *Assuming the SXDH assumption (listed in Conjecture 3.14) holds for groups* G_X, G_Y, G_T*, a bilinear map* $e : G_X \times G_Y \to G_T$ *is not a secure weak 2-immunizer.*

OPEN QUESTION. Summarizing, our results largely settle the feasibility of designing secure 2-immunizers for backdoor PRGs, but leave the following fascinating question open: *Is there a* 2-*immunizer* C *in the standard model whose security can be black-box reduced to an efficiently falsifiable assumption?*

While we know such C cannot be "highly dependent on both inputs", which rules out most natural choices one would consider (including cryptographic hash function), we do not know if other "unnatural" functions C might actually work.

In the absence of such a function/reduction, there are two alternatives:

First, it may be possible to give a *non-black-box* reduction from a non-highly input-dependent function (such as a very good two-source extractor).

Or alternatively, one might try to base the security of C on a *non-falsifiable* assumption likely satisfied by a real-world cryptographic hash function. For example, [21] built seeded 1-immunizers based on the existence of so called universal computational extractors (UCEs) [8]. Unfortunately, the UCE definition seems to be inherently fitted for 1-immunizers, and it is unclear (and perhaps unlikely) that something similar can be done in the 2-immunizer setting, at least with a security definition that is noticeably simpler than that of 2-immunizers.

1.4 Further Related Work

We briefly mention several related works not mentioned so far.

Extractors. Randomness Extractors convert a weak random source into an output which is statistically close to uniform. Similar to our setting, while deterministic extraction is impossible in this generality [17], these results can either be overcome using seeded extractors [31], or two-source extractors [17].

A special class of seeded extractors consider consider sources which could partially depend on the prior outputs of the extractor (and, hence, indirectly on the random seed). Such sources are called *extractor-dependent* [24, 33], and generalize the corresponding notion of oracle-dependent extractors considered by [19] in the ROM. Conceptually similar to our results, [24] showed a black-box separation for constructing such extractors from cryptographic hash functions in the standard model, despite the fact that cryptographic hash functions provably worked in the ROM [19].

Kleptography. Young and Yung studied what they called kleptography: subversion of cryptosystems by modifying encryption algorithms in order to leak information subliminally [45, 46, 47]. Juels and Guajardo [29] propose an immunization scheme for kleptographic key-generation protocols that involves publicly-verifiable injection of private randomness by a trusted entity. More recent work by Bellare, Paterson, and Rogaway [9] treats a special case of Young and Yung's setting for symmetric encryption.

As described in detail above, the works [37, 36] consider the idea of using a random oracle as a 2-immunizer, however their results do not extend to the stateful setting considered here.

The works [34, 5] also consider immunizing corrupted PRGs, however these results success by modifying the public parameters, as opposed to operating on the PRG output. In other words, the immunizers are not simple and stateless, and thus not relevant in a situations where a user cannot control the implementation itself (e.g. if it is implemented in hardware or the kernel).

Steganography and Related Notions. Steganography (see [40, 27]) is the problem of sending a hidden message in communications over a public channel so that an adversary eavesdropping on the channel cannot even detect the presence of the hidden message. In this sense backdoor PRG could be viewed as a steganographic channel where the PRG is trying to communicate information back to the malicious PRG designer, without the "public" being able to detect such communication (thinking instead that a random stream is transmitted).

More recently, the works of [28, 32] looked at certain types of encryption schemes which can always be turned into stegonagraphic channels, even if the dictator demands the users to reveal their purported secret keys.

Finally, the works of [30, 23] looked at constructing so called *reverse firewalls*, which probably remove steganographic communication by carefully re-randomizing messages supposedly exchanged by the parties for some other cryptographic task.

Backdoored Random Oracles. The work of [7] and [11] consider the task of immunizing random oracles with XOR. However, these consider information theoretic models of PRG security. An intriguing observation about the findings of our work is that information theoretic models (such as the backdoored random oracle model) do not capture the computational advantage that backdoors can achieve, as is shown by our counterexamples in section 3.

2 Definitions

Definition 2.1. *Two distributions X and Y are called (t, ϵ)-indistinguishable (denoted by $\boldsymbol{CD}_t(X, Y) \leq \epsilon$) if for any algorithm D running in time t,*

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \leq \epsilon.$$

Definition 2.2. *Let X_λ and Y_λ be two families of distributions indexed by λ. If for all polynomial $t(\lambda)$ and some negligible $\epsilon(\lambda)$, X_λ and Y_λ are $(t(\lambda), \epsilon(\lambda))$-indistinguishable, then we say X and Y are computationally indistinguishable (denoted by $\boldsymbol{CD}(X, Y) \leq negl(\lambda)$).*

2.1 Pseudorandom Generators

A pseudorandom generator is a pair of algorithms (K, G). Traditionally, K takes in randomness and outputs a public parameter. We additionally allow K to output a secret key to be used for defining trapdoors. To go with our notation of secret keys, we will denote the public parameter as the public key. For non-trapdoored PRGs, the secret key is set to null. G is a function that takes in a public key and a state, and outputs an n-bit output as well as a new state. More formally, we give the following definitions, adapted from [21]:

Definition 2.3. *Let $\mathcal{PK}, \mathcal{SK}$ be sets of public and secret keys respectively. Let $\mathcal{S}$ be a set we call the state space. A pseudorandom generator (PRG) is a pair of algorithms (K, G) where*

- $K : \{0,1\}^\ell \to \mathcal{PK} \times \mathcal{SK}$ takes in randomness and outputs a public key pk and secret key sk. We will denote running K on uniform input as $(pk, sk) \xleftarrow{\$} K$.

- $G : \mathcal{PK} \times \mathcal{S} \to \{0,1\}^n \times \mathcal{S}$ takes in the public key and a state and outputs n bits as well as the new state.

For ease of notation, we may write G instead of G_{pk} when the public key is clear from context.

Definition 2.4. *Let (K, G) be a PRG, $pk \in \mathcal{PK}, s \in \mathcal{S}$. Let $s_0 = s$ and let $(r_i, s_i) \leftarrow G_{pk}(s_i)$ for $i \geq 1$. We call the sequence $(r_1, \ldots, r_q)$ the output of (K, G), and denote it by $\boldsymbol{out}^q(G_{pk}, s)$ (or $\boldsymbol{out}^q(G, s)$).*

For n an integer we will denote by $\mathcal{U}_n$ the uniform distribution over $\{0,1\}^n$.

Definition 2.5. *A PRG (K, G) is a (t, q, δ) publicly secure PRG if K, G both run in time t and*

$$pk \xleftarrow{\$} K$$

$$\boldsymbol{CD}_t((pk, \boldsymbol{out}^q(G_{pk}, \mathcal{S})), (pk, \mathcal{U}_{qn})) \leq \delta.$$

Note that here there is some implied initial distribution over $\mathcal{S}$. This will depend on the construction, but when unstated we will assume that this distribution is uniform.

Definition 2.6. *A PRG (K, G) is a (t, q, δ) backdoor secure PRG if K, G both run in time t and*

$$(pk, sk) \xleftarrow{\$} K$$

$$\boldsymbol{CD}_t((pk, sk, \boldsymbol{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})), (pk, sk, \mathcal{U}_{qn})) \leq \delta.$$

Note that there are PRGs that are (t, q, δ) publicly secure, but not (t', q, δ') backdoor secure even for some $t' << t$ and $\delta' >> \delta$ [21]. The goal of an immunizer is to take in as input some (K, G) which is publicly secure but not backdoor secure, and transform it generically into a new PRG which is backdoor secure.

2.2 2-Immunizers

Our definition of 2-immunizers will also be based on the definition of immunizers given in [21]. Note in particular that while the [21] definition of immunizers takes in the output of one PRG and a random seed, we define 2-immunizers to be deterministic functions of the output of two PRGs.

We first define notation to express what it means to apply an immunizer to two PRGs.

Definition 2.7. *Let $(K^X, G^X), (K^Y, G^Y)$ be two PRGs and let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a function on the output spaces of the PRGs. We define a new PRG as follows:*

-The key generation algorithm (denoted (K^X, K^Y)) will be the concatenation of the original two key generation algorithms. More formally, it will run $K^X \to pk^X, sk^X$, $K^Y \to pk^Y, sk^Y$ and will return $pk = (pk^X, pk^Y)$ and $sk = (sk^X, sk^Y)$.

-The pseudorandom generation algorithm, denoted $C(G^X, G^Y)$ will run both PRGs independently and apply C to the output. Formally, let us denote $s = (s^X, s^Y)$. If $G^X(s^X) = (r^X, s'^X)$ and $G^Y(s^Y) = (r^Y, s'^Y)$, then

$$C(G^X, G^Y)(s) := (C(r^X, r^Y), (s'^X, s'^Y)).$$

Note that the output of the PRG will be C applied to the outputs of the original PRGs. Formally, if $\mathbf{out}^q(G^X, s^X) = x_1, \ldots, x_q$ and $\mathbf{out}^q(G^Y, s^Y) = y_1, \ldots, y_q$, then

$$\mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y)) = C(x_1, y_1), \ldots, C(x_q, y_q).$$

Definition 2.8. *A two-input function C is a (t, q, δ, δ')-secure weak 2-immunizer, if for any (t, q, δ) publicly secure PRGs $(K^X, G^X), (K^Y, G^Y)$, the PRG $((K^X, K^Y), C(G^X, G^Y))$ is a (t, q, δ') backdoor secure PRG.*

A weak 2-immunizer is effective at immunizing two PRGs as long as the public parameters are independently sampled. We can also consider the case where the designers of the two PRGs collude and share public parameters. Identically, we can consider the case where we run one backdoored PRG on multiple honest initializations. If a 2-immunizer effectively immunizes in this setting, we call it a strong 2-immunizer.

Let us first define the syntax

Definition 2.9. *Let (K, G) be a PRG and let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a function on the output space of G. We define a new PRG (denoted $(K, C(G, G))$) as follows:*

-The key generation algorithm will be K

-The pseudorandom generation algorithm, denoted $C(G_{pk}, G_{pk})$ will run G twice (with the same public key) on two initial seeds, and apply C to the output. Formally, let us denote $s = (s^X, s^Y)$. If $G_{pk}(s^X) = (r^X, s'^X)$ and $G_{pk}(s^Y) = (r^Y, s'^Y)$, then

$$C(G, G)(s) := (C(r^X, r^Y), (s'^X, s'^Y))$$

If $x_1, \ldots, x_q = \mathbf{out}^q(G_{pk}, s^X)$ and $y_1, \ldots, y_q = \mathbf{out}^q(G_{pk}, s^Y)$ are two outputs of G on the same public key and freshly sampled initial states, then

$$\mathbf{out}^q(C(G,G),(s^X,s^Y)) = C(x_1,y_1),\ldots,C(x_q,y_q).$$

Definition 2.10. *A two-input function C is called a (t,q,δ,δ')-secure strong 2-immunizer, if for any (t,q,δ) publicly secure PRG (K,G), the PRG $(K, C(G,G))$ is a (t,q,δ') backdoor secure PRG.*

Lemma 2.11. *If C is a (t,q,δ,δ')-secure strong 2-immunizer, then C is a $(t,q,\delta,4\delta')$-secure weak 2-immunizer.*

Proof. Let $(K^X,G^X),(K^Y,G^Y)$ be two (t,q,δ) publicly secure PRGs, and let C be a (t,q,δ,δ')-secure strong 2-immunizer. Let $pk = (pk^X, pk^Y)$ and $sk = (sk^X, sk^Y)$. We need to show that

$$\mathbf{CD}_t((pk,sk,\mathbf{out}^q(C(G^X,G^Y),(\mathcal{S}^X,\mathcal{S}^Y)),(pk,sk,\mathcal{U}_{qm})) \leq 4\delta'$$

We will construct a new PRG (K,G) which runs K^X, G^X and K^Y, G^Y in parallel and chooses which to use based on the first bit of the state. More formally,

- K: Run $K^X \to (pk^X, sk^X)$ and $K^Y \to (pk^Y, sk^Y)$. Output $pk = (pk^X, pk^Y)$ and $sk = (sk^X, sk^Y)$.

- $G_{pk^X,pk^Y}(s)$: Parse s as (b, s^X, s^Y) with $b \in \{X,Y\}$, $s^X \in \mathcal{S}^X$, $s^Y \in \mathcal{S}^Y$. Run $G^b_{pk^b}(s^b) \to (r', s')$, and output $(r', (b, s'))$. We will denote the state space $\mathcal{S} := \{X,Y\} \times \mathcal{S}^X \times \mathcal{S}^Y$.

Since C is a (t,q,δ,δ')secure strong 2-immunizer, we have

$$(pk,sk) \xleftarrow{\$} K$$

$$\mathbf{CD}_t((pk,sk,\mathbf{out}^q(C(G,G),(\mathcal{S},\mathcal{S}))),(pk,sk,\mathcal{U}_{qm})) \leq \delta'.$$

Note that given any two samples $(b,s),(b',s')$ from $\mathcal{S}$, with probability $\frac{1}{4}$, we will have $b = X$ and $b' = Y$. Conditioned on this event, by construction we have that $(pk, sk, \mathbf{out}^q(C(G_{pk}, G_{pk}), (\mathcal{S},\mathcal{S}))$ is distributed identically to $((pk^X, pk^Y), (sk^X, sk^Y), \mathbf{out}^q(C(G^X,G^Y),(\mathcal{S}^X,\mathcal{S}^Y))$. Thus, we have

$$(pk^X,sk^X) \xleftarrow{\$} K^X, (pk^Y,sk^Y) \xleftarrow{\$} K^Y, pk = (pk^X,pk^Y), sk = (sk^X,sk^Y)$$

$$\mathbf{CD}_t((pk,sk,\mathbf{out}^q(C(G^X,G^Y),(\mathcal{S}^X,\mathcal{S}^Y)),(pk,sk,\mathcal{U}_{qm})) \leq 4\delta'.$$

□

Remark 2.12. Some traditional definitions of PRGs [10] consider the notion of forward-secrecy. That is, even PRG security for the first q outputs should still be maintained even if the $q+1$st output is leaked. However, it is impossible for a 2-immunizer in our model to preserve public forward secrecy. Informally, given any PRG satisfying forward-secrecy, we can append an encryption of the initial state to the $q+1$st state. This would result in a PRG satisfying public forward-secrecy but not backdoor forward-secrecy. Since we do not allow the 2-immunizer to view or modify the internal state of the corresponding PRGs in any way, it is impossible for any 2-immunizer to remove this vulnerability.

3 Counterexamples for Simple 2-Immunizers

In this section we will outline a framework for arguing that simple functions (for example XOR) do not work as 2-immunizers. To argue that some C is not a strong 2-immunizer, we will construct a public key encryption scheme suitably homomorphic under C. We will then note that the PRG which simply encrypts 0 using the randomness of its honest initialization will have a backdoor after immunization, where the backdoor will be given by the homomorphic property of the underlying encryption scheme.

To argue that C is not a weak 2-immunizer, we will need to instead construct two public key encryption schemes which are in jointly homomorphic in a suitable manner. In this case, the PRGs defined by encrypting 0 under the two public key encryption schemes defined will allow us to perform an analogous attack on C.

In particular, we will generically define what it means for public key encryption schemes to be suitably homomorphic under C, and argue that this property is enough to show that C is not a 2-immunizer. Note that the definition of suitably homomorphic will depend on whether we are attacking the weak or strong security of C.

We will then instantiate our generic result with specific public key encryption schemes, leading to the following theorems.

Theorem 3.1 (Theorem 1.2 restated)**.** *Assuming the Alekhnovich assumption (listed in Proposition 3.7) holds, XOR is not a* $(poly(\lambda), 1, negl(\lambda), negl(\lambda))$*-secure strong 2-immunizer.*

Note that there is no simple way to adapt the public key encryption scheme used to prove this theorem to be sufficiently homomorphic to prove that XOR is not a weak 2-immunizer. We leave the question as to whether XOR is a weak 2-immunizer as an open question.

Definition 3.2. *Let* G_X, G_Y, G_T *be groups of prime order exponential in* λ *with generators* g_X, g_Y, g_T*. A bilinear map* $e : G_X \times G_Y \to G_T$ *is a function satisfying*

$$e(g_X^a, g_Y^b) = e(g_X, g_Y)^{ab} = g_T^{ab}$$

Note that requiring $e(g_X, g_Y) = g_T$ is a non-standard requirement for bilinear maps, but will always occur when we restrict the codomain of the bilinear group to the subgroup defined by its image.

Theorem 3.3 (Theorem 1.5 restated)**.** *Assuming the SXDH assumption (listed in Conjecture 3.14) holds for groups* G_X, G_Y, G_T*, a bilinear map* $e : G_X \times G_Y \to G_T$ *is not a* $(poly(\lambda), 2, negl(\lambda), negl(\lambda))$*-secure weak 2-immunizer.*

Note that although [7] does not directly argue that a bilinear map is a 2-immunizer in their model, it is clear that the argument for XOR can be generalized to apply for bilinear maps.

3.1 Public Key Encryption

A public key encryption scheme (PKE) is a triple $(\text{Gen}, \text{Enc}, \text{Dec})$ where

- Gen outputs a public key, secret key pair (pk, sk),

- Enc takes in the public key pk and a message m, and outputs a ciphertext c,

- Dec takes in the secret key sk and a ciphertext c, and outputs the original message m.

For security, as we are working with pseudorandom generators, it is useful for us to require that the encryption schemes themselves be pseudorandom. More formally,

Definition 3.4. *We say that a public key encryption scheme* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is pseudorandom if for all* m*,*

$$pk \xleftarrow{\$} \text{Gen}$$
$$\boldsymbol{CD}_{poly(\lambda)}((pk, \text{Enc}(m)), (pk, \mathcal{U})) \leq negl(\lambda)$$

Note that for our purposes we will require all public key encryption schemes to be pseudorandom. We remark that this assumption is strictly stronger than traditional PKE security.

3.2 Strong 2-Immunizers

Definition 3.5. *Let* $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *be some operation. We say that a public key encryption scheme* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is* $\underline{C\text{-homomorphic}}$ *if there exists some function* Dec_{sk}^C *such that for all* m*,*

$$\Pr_{\substack{(pk,sk)\xleftarrow{\$}\text{Gen} \\ \alpha,\alpha'\xleftarrow{\$}\mathcal{U}}} [\text{Dec}_{sk}^C(C(\text{Enc}_{pk}(m;\alpha), \text{Enc}_{pk}(m;\alpha'))) = m] \geq \frac{2}{3}.$$

Theorem 3.6. *Let* $(\text{Gen}, \text{Enc}, \text{Dec})$ *be a public key encryption scheme and let* C *be some operation. Then, if* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is pseudorandom and* C*-homomorphic (with homomorphic decryption algorithm* Dec^C*), then* C *is not a* $(poly(\lambda), 1, negl(\lambda), negl(\lambda))$*-secure strong 2-immunizer.*

Proof. We will first construct a PRG (K, G) using $(\text{Gen}, \text{Enc}, \text{Dec})$, and then we will show that $C(G, G)$ has a backdoor.

Let us first observe that $\Pr[\text{Dec}^C(\mathcal{U}) \to 0] + \Pr[\text{Dec}^C(\mathcal{U}) \to 1] \leq 1$, and so one of these probabilities will be less than $\frac{1}{2}$. Without loss of generality, assume $\Pr[\text{Dec}^C(\mathcal{U}) \to 0] \leq \frac{1}{2}$.

Define (K, G) by $K := \text{Gen}$, $G_{pk}(s) := \text{Enc}_{pk}(0; s)$. It is clear to see that (K, G) is a $(poly(\lambda), 1, negl(\lambda))$ publicly secure PRG by the definition of a pseudorandom PKE. Thus, it remains to show an adversary D that can distinguish

$$(pk, sk, C(\text{Enc}_{pk}(0;\mathcal{U}), \text{Enc}_{pk}(0;\mathcal{U})))$$

from

$$(pk, sk, \mathcal{U})$$

with probability $\geq \frac{1}{poly(\lambda)}$.

On input (pk, sk, r), D will run $\text{Dec}_{sk}^C(r) \to m$ and output 1 if and only if $m = 0$. It is clear that

$$\Pr[D(pk, sk, C(\text{Enc}_{pk}(0;\mathcal{U}), \text{Enc}_{pk}(0;\mathcal{U}))) \to 1] \geq \frac{2}{3}$$

by the definition of Dec^C. But note that we assumed $\Pr[\mathrm{Dec}^C(\mathcal{U}) \to 0] \leq \frac{1}{2}$, and so

$$\Pr[D(pk, sk, \mathcal{U}) \to 1] \leq \frac{1}{2}$$

Thus, the advantage of D is $\geq \frac{2}{3} - \frac{1}{2} = \frac{1}{6} \geq \frac{1}{poly(\lambda)}$ □

We remark that while this theorem is stated for $q = 1$, it is fairly easy to extend this to arbitrary q by simply appending the corrupted PRGs with a genuine one.

[3] gives a construction of a public key encryption scheme based off of a variant of the learning parity with noise problem (which we will call the Alekhnovich assumption, it is Conjecture 4.7 in his paper). Instead of presenting his underlying assumption directly, we will refer to the following proposition:

Proposition 3.7. *[3]: Suppose that the Alekhnovich assumption holds, then for every $m = O(n), k = \Theta(\sqrt{n})$, $\ell, t \leq poly(n)$ then*

$$A_i \xleftarrow{\$} \mathcal{U}_{m \times n}, x_i \xleftarrow{\$} \mathcal{U}_n, e_i \xleftarrow{\$} \binom{\{0,1\}^m}{k}$$

$$\boldsymbol{CD}_t((A_i, A_i x_i + e_i)_{i=1}^{\ell}, (A_i, \mathcal{U}_m)_{i=1}^{\ell}) \leq negl(n)$$

That is, given a uniformly random $m \times n$ binary matrix A, a vector which differs from an element in the image of the matrix in exactly k places is computationally indistinguishable from random.

Let us proceed now to the proof of Theorem 3.1.

We will prove this by showing a pseudorandom $\oplus$-homomorphic public key encryption scheme based off of the Alekhnovich assumption.

We claim that if the Alekhnovich assumption holds, the public key encryption scheme presented in [2] (along with a minor variation) is both pseudorandom and $\oplus$-homomorphic. Therefore, by Theorem 3.6, XOR is not a strong 2-immunizer.

First, we present Alekhnovich's public key encryption scheme in Figure 1. We make one minor change to the original scheme, namely we change the value of the parameter k from $\sqrt{\frac{n}{2}}$ to $\sqrt{\frac{n}{4}}$. Note that since the underlying proposition only requires that $k = \Theta(\sqrt{n})$, this does not affect the proof of security.

Proposition 3.8. *[3]: Assuming the Alekhnovich assumption holds,*

$$\boldsymbol{CD}((pk, \text{Enc-A}(0)), (pk, \text{Enc-A}(1))) \leq negl(\lambda)$$

Corollary 3.9. *Assuming the Alekhnovich assumption holds,*
(Gen-A, Enc-A, Dec-A) *is pseudorandom.*

Proposition 3.10. *Assuming the Alekhnovich assumption holds,*
(Gen-A, Enc-A, Dec-A) *as presented above is $\oplus$-homomorphic.*

Notation:
$k = \sqrt{\frac{n}{4}}$, $m = 2n$.
$\{0,1\}^\ell$ are vectors in $\mathbb{Z}_2^\ell$.
$\binom{\{0,1\}^m}{k}$:= vectors in $\{0,1\}^m$ with exactly k 1s.

Gen-A:
$A \xleftarrow{\$} \mathcal{U}_{m\times n}$.
$x \xleftarrow{\$} \mathcal{U}_n$
$e \xleftarrow{\$} \binom{\{0,1\}^m}{k}$
$b \leftarrow Ax + e$, $M = (b|A)$.
$B \xleftarrow{\$} \mathcal{U}_{m\times(m-n-1)}$ conditioned on $M^T B = 0_n$.
Output $pk = B$, $sk = (B, e)$.

Enc-A(1):
$c \xleftarrow{\$} \mathcal{U}_m$.
Output c.

Enc-A(0):
$x' \xleftarrow{\$} \mathcal{U}_{n-1}$,
$e' \xleftarrow{\$} \binom{\{0,1\}^m}{k}$.
Output $c = Bx' + e'$.

Dec-A($(B, e), c$):
Output 0 if $e^T c = 0$.
Otherwise, output 1.

Figure 1: Alekhnovich's PKE scheme (From Section 4.4.3).

Proof. We simply need to show that there exists some Dec-A$^\oplus$ decrypting the sum of any two messages. Set Dec-A$^\oplus$ = Dec-A.

Observe that addition over $\mathbb{Z}_2^\ell$ is equivalent to $\oplus$, and so we will denote $\oplus$ as $+$.

Since $\Pr[\text{Dec-A}(\text{Enc-A}(1)) = 0] = \frac{1}{2}$, and the sum of two uniformly random values is itself uniformly random, $\Pr[\text{Dec-A}(\text{Enc-A}(1) + \text{Enc-A}(1)) = 0] = \frac{1}{2}$. Thus, if we can show that $\Pr[\text{Dec-A}(\text{Enc-A}(0) + \text{Enc-A}(0)) = 0] \geq \frac{1}{2} + c$ for some constant c, we are done.

Let $c_1, c_2 \xleftarrow{\$} \text{Enc-A}(0)$. We can write $c_1 = Bx_1 + e_1$ and $c_2 = Bx_2 + e_2$.

$$\text{Dec}(c_1 + c_2) = e^T(Bx_1 + e_1 + Bx_2 + e_2) = e^T B(x_1 + x_2) + e^T e_1 + e^T e_2$$

Following the proof of correctness from [3], we let p_i be the probability that e_b misses the first i non-zero entries of e. Hence, the probability (over the choise of e_b) that e and e_b have a common one can be written as

$$\sum_{i=0}^{k-1} p_i \frac{k}{m-i} < \frac{k^2}{m-k} < \frac{k^2}{m-k^2} = \frac{1}{7}$$

Thus, the probability that e has a common one with either e_1 or e_2 is $\leq \frac{2}{7}$. Therefore, $\Pr[\text{Dec-A}(\text{Enc-A}(0)+$ $\text{Enc-A}(0)) = 1] \leq \frac{2}{7}$, and so $\Pr[\text{Dec-A}(\text{Enc-A}(0) + \text{Enc-A}(0)) = 1] \geq \frac{1}{2} + \frac{3}{14}$. □

3.3 Weak 2-Immunizers

Definition 3.11. *Let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be some operation. We say a pair of public key encryption schemes $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ and $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ are jointly C-homomorphic if there exists some function $\mathrm{Dec}^C_{sk,sk'}$ such that for all m,*

$$\Pr_{\substack{(pk,sk)\xleftarrow{\$}\mathrm{Gen} \\ (pk',sk')\xleftarrow{\$}\mathrm{Gen}' \\ \alpha,\alpha'\xleftarrow{\$}\mathcal{U}}} [\mathrm{Dec}^C_{sk,sk'}(C(\mathrm{Enc}_{pk}(m;\alpha), \mathrm{Enc}'_{pk'}(m;\alpha'))) = m] \geq \frac{2}{3}.$$

Theorem 3.12. *Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ be two public key encryption schemes and let C be some operation. Then, if $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ are pseudorandom and jointly C-homomorphic (with homomorphic decryption algorithm Dec^C), then C is not a $(poly(\lambda), 1, negl(\lambda), negl(\lambda))$-secure weak 2-immunizer.*

Proof. This proof is analogous to the proof of Theorem 3.6. The corresponding PRGs are $(K^X, G^X) = (\mathrm{Gen}, \mathrm{Enc}(0;s))$ and $(K^Y, G^Y) = (\mathrm{Gen}', \mathrm{Enc}'(0;s))$. The distinguisher again runs $\mathrm{Dec}^C \to 0$ and returns 1 if and only if $m = 0$. □

Corollary 3.13. *If there exists $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ pseudorandom and jointly $\oplus$-homomorphic, then $\oplus$ is not a $(poly(\lambda), 1, negl(\lambda), negl(\lambda))$-secure weak 2-immunizer.*

We remark that the Alekhnovich PKE is not jointly $\oplus$-homomorphic with itself. We leave it as an open question as to whether such a pair of encryption schemes exist for XOR, but we suspect that its existence is likely.

Instead, we show that another simple 2-immunizer (namely a bilinear pairing) is not secure assuming a suitable computational assumption. In particular, we will rely on the SXDH assumption, defined in [6, 4].

Conjecture 3.14. *The Symmetric External Diffie Hellman Assumption (SXDH) states that there exist groups G_X, G_Y, G_T with generators g_X, g_Y, g_T such that -there exists an efficiently computable bilinear map $e : G_X \times G_Y \to G_T$, -for uniformly random $a, b, c \xleftarrow{\$} \mathbb{Z}_{|G_X|}$ $\boldsymbol{CD}((g_X^a, g_X^b, g_X^{ab}), (g_X^a, g_X^b, g_X^c)) \leq negl(\lambda)$ (the Diffie Hellman assumption holds for G_X), -for uniformly random $a, b, c \xleftarrow{\$} \mathbb{Z}_{|G_Y|}$ $\boldsymbol{CD}((g_Y^a, g_Y^b, g_Y^{ab}), (g_Y^a, g_Y^b, g_Y^c)) \leq negl(\lambda)$ (the Diffie Hellman assumption holds for G_Y).*

Note that, as stated in Definition 3.2 we will require that $e(g_X, g_Y) = g_T$ and that G_X, G_Y, G_T are of prime order exponential in λ.

Note that instead of constructing jointly homomorphic public key encryption schemes under e, we will instead create public key encryption schemes jointly homomorphic under a related operation. We will then use the fact that this related operation is not a weak 2-immunizer to show that e is not a weak 2-immmunizer.

Let G_X, G_Y, G_T be cyclic groups of size exponential in λ with an efficiently computable bilinear map $e : G_X \times G_Y \to G_T$. Define the 2-immunizer $C^e : (G_X \times G_X) \times (G_Y \times G_Y) \to G_T$ by

$$C^e((a_X, b_X), (a_Y, b_Y)) = (e(a_X, b_X), e(a_Y, b_Y)).$$

Lemma 3.15. *Assuming the SXDH assumption holds, C^e is not a $(poly(\lambda), 1, negl(\lambda), negl(\lambda))$-secure weak 2-immunizer.*

We defer the proof of this lemma to Section 3.4.

By breaking up the output of the counterexample for Lemma 3.15 into two parts, we get a counterexample proving Theorem 3.3. Formally, we prove Theorem 3.3 as follows:

Proof. By Lemma 3.15, we know that there exists $(K^X, G^X), (K^Y, G^Y)$ such that $C^e(G^X, G^Y)$ is not a $(poly(\lambda), 1, negl(\lambda))$ backdoor-secure PRG. But we can write $G^X(s) = (a^X, b^X)$ and $G^Y(s) = (a^Y, b^Y)$. Define

$$K'^X = (K^X, 0), G'^X(s, 0) = (a^X, (s, 1)), G'^X(s, 1) = b^X$$
$$K'^Y = (K^Y, 0), G'^Y(s, 0) = (a^Y, (s, 1)), G'^Y(s, 1) = b^Y$$

That is, G'^X, G'^Y are PRGs defined by breaking up the outputs of G^X, G^Y into two different rounds. It is clear that

$$\mathbf{out}^2(e(G'^X, G'^Y), ((s^X, 0), (s^Y, 0)))$$

is identically distributed to

$$\mathbf{out}^1(C^e(G^X, G^Y), (s^X, s^Y))$$

and so clearly $e(G'^X, G'^Y)$ is not a $(poly(\lambda), 2, negl(\lambda))$ backdoor-secure PRG. □

3.4 Proof of Lemma 3.15

We will construct two public key encryption schemes such that they are jointly C^e-homomorphic. These encryption schemes, denoted
$(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ and $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ are denoted in Figure 2

Proposition 3.16. *Assuming the SXDH assumption holds,* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *and* $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ *are pseudorandom.*

Proof. We will show that $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is pseudorandom and the same argument will apply for $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$.

It is clear that

$$\mathbf{CD}((pk, \text{Enc-A}(1)), (pk, \mathcal{U})) \leq negl(\lambda).$$

Also, $\mathrm{Enc}(0) = (g_X^{r_X}, (g_X^{s_X})^{r_X})$ and we know from the SXDH assumption that $\mathbf{CD}((g_X^a, g_X^b, g_X^{ab}), (g_X^a, g_X^b, g_X^c)) \leq negl(\lambda)$. Thus,

$$\mathbf{CD}((g_X^{s_X}, g_X^{r_X}, g_X^{s_X r_X}), (g_X^{s_X}, g_X^{r_X}, g_X^{r'_X})) \leq negl(\lambda)$$

and so we are done. □

Proposition 3.17. *Assuming the SXDH assumption holds,* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *and* $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ *are jointly C^e-homomorphic.*

Gen:	Gen′:
$s_X \xleftarrow{\$} \mathbb{Z}_{\lvert G_X \rvert}$.	$s_Y \xleftarrow{\$} \mathbb{Z}_{\lvert G_Y \rvert}$.
Output $pk = g_X^{s_X}$, $sk = s_X$.	Output $pk = g_Y^{s_Y}$, $sk = s_Y$.
$\mathrm{Enc}_{g_X^{s_X}}(0)$:	$\mathrm{Enc}'_{g_Y^{s_Y}}(0)$:
$r_X \xleftarrow{\$} \mathbb{Z}_{\lvert G_X \rvert}$.	$r_Y \xleftarrow{\$} \mathbb{Z}_{\lvert G_Y \rvert}$.
$c_1 \leftarrow g_X^{r_X}$,	$c_1 \leftarrow g_Y^{r_Y}$,
$c_2 \leftarrow (g_X^{s_X})^{r_X}$.	$c_2 \leftarrow (g_Y^{s_Y})^{r_Y}$.
Output (c_1, c_2).	Output (c_1, c_2).
$\mathrm{Enc}_{g_X^{s_X}}(1)$:	$\mathrm{Enc}'_{g_Y^{s_Y}}(1)$:
$r_X, r'_X \xleftarrow{\$} \mathbb{Z}_{\lvert G_X \rvert}$.	$r_Y, r'_Y \xleftarrow{\$} \mathbb{Z}_{\lvert G_Y \rvert}$.
$c_1 \leftarrow g_X^{r_X}$, $c_2 = g_X^{r'_X}$.	$c_1 \leftarrow g_Y^{r_Y}$, $c_2 = g_Y^{r'_Y}$.
Output (c_1, c_2).	Output (c_1, c_2).
$\mathrm{Dec}_{s_X}(c_1, c_2)$:	$\mathrm{Dec}'_{s_Y}(c_1, c_2)$:
Output 1 if and only if $c_1^{s_X} = c_2$.	Output 1 if and only if $c_1^{s_Y} = c_2$.

Figure 2: Jointly homomorphic PKE schemes under repeated pairings.

Proof. We will define $\mathrm{Dec}^{C^e}_{s_X,s_Y}(c_1, c_2)$ to output 1 if and only if $c_1^{s_X s_Y} = c_2$.

Then

$$\begin{aligned} \mathrm{Dec}^{C^e}_{s_X,s_Y}(C^e(\mathrm{Enc}(0), \mathrm{Enc}'(0))) &= \mathrm{Dec}^{C^e}(C^e((g_X^{r_X}, g_X^{s_X r_X}), (g_Y^{r_Y}, g_Y^{s_Y r_Y}))) \\ &= \mathrm{Dec}^{C^e}(e(g_X, g_Y)^{r_X r_Y}, e(g_X, g_Y)^{r_X r_Y s_X s_Y}) \\ &= 1 \end{aligned} \tag{1}$$

and so $\Pr[\mathrm{Dec}^{C^e}(C^e(\mathrm{Enc}(0), \mathrm{Enc}'(0))) \to 0] = 1$.

Also,

$$\begin{aligned} \mathrm{Dec}^{C^e}_{s_X,s_Y}(C^e(\mathrm{Enc}(0), \mathrm{Enc}'(0))) &= \mathrm{Dec}^{C^e}(C^e((g_X^{r_X}, g_X^{r'_X}), (g_Y^{r_Y}, g_Y^{r'_Y}))) \\ &= \mathrm{Dec}^{C^e}(e(g_X, g_Y)^{r_X r_Y}, e(g_X, g_Y)^{r'_X r'_Y}). \end{aligned} \tag{2}$$

Thus,

$$\Pr[\mathrm{Dec}^{C^e}(C^e(\mathrm{Enc}(1), \mathrm{Enc}'(1))) \to 0] = \Pr[e(g_X, g_Y)^{r_X r_Y s_X s_Y} = e(g_X, g_Y)^{r'_X r'_Y}]$$

But note that exponentiation by any non-trivial scalar is a permutation on G_T, and so we have

$$\Pr[\mathrm{Dec}^{C^e}(C^e(\mathrm{Enc}(1), \mathrm{Enc}'(1))) \to 0] = \Pr[e(g_X, g_Y)^{r} = e(g_X, g_Y)^{r'}]$$

But since $e(g_X, g_Y)$ is a generator, this occurs with probability $\dfrac{1}{|G_T|} = negl(\lambda)$. Therefore,

$$\Pr[\mathrm{Dec}^{C^e}(C^e(\mathrm{Enc}(1), \mathrm{Enc}'(1))) \to 1] = 1 - negl(\lambda).$$

□

By Theorem 3.12, we have proven Lemma 3.15.

4 Positive Result in Random Oracle Model

Although it seems that simple functions will not function well as a 2-immunizer, we show that a random oracle is a strong 2-immunizer. Heuristically, this means that a good hash function can be used in practice as a 2-immunizer. Furthermore, it gives some hope that 2-immunizers may exist in the standard model.

In fact, a random oracle is a strong 2-immunizer even if we allow the adversary to perform arbitrary preprocessing on the random oracle. This model, introduced in [41], is known as the Auxiliary Input Random Oracle Model (AI-ROM).

Theorem 4.1. *Let $RO : \{0,1\}^{2n} \to \{0,1\}^m$ be a random oracle. For t sufficiently large to allow for simple computations, $f(X,Y) = RO(X||Y)$ is a (t, q, δ, δ')-secure strong 2-immunizer with*

$$\delta' = \left(\delta + \frac{q^2}{2^n}\right) + 2(t+t^2)q\sqrt{\delta + \frac{q}{2^n}}.$$

Corollary 4.2. *$f(X,Y) = RO(X||Y)$ is a $(poly(\lambda), poly(\lambda), negl(\lambda), negl(\lambda))$-secure strong 2-immunizer in the ROM.*

Theorem 4.3 (Theorem 1.3 restated)**.** *$f(X,Y) = RO(X||Y)$ is a $(poly(\lambda), poly(\lambda), negl(\lambda), negl(\lambda))$-secure strong 2-immunizer in the AI-ROM.*

The intuition behind Theorem 4.1 is as follows. Even given the secret and public keys for a PRG, public security guarantees that the output of each PRG is unpredicable. Let $x_1, \ldots, x_q$ and $y_1, \ldots, y_q$ be two outputs of a PRG, and let us consider the perspective of the compromised PRG generating x. Since this algorithm does not know the seed generating y, each y_i is unpredictable to it. Thus, it has no way of seeing any of the outputs of the functions $RO(\cdot||y_i)$. But as long as neither call to the PRG queries the random oracle on $x_i||y_i$, there will be no detectable relationship between the x_i's and $RO(x_i||y_i)$, and so the immunizer output will seem truly random.

The extension to the AI-ROM in Theorem 4.3 comes from standard presampling techniques.

4.1 Random Oracle Model Definitions

In the random oracle model (ROM), we treat some function RO as a function chosen uniformly at random. This provides a good heuristic for security when the random oracle is instantiated with some suitable hash function. To argue that some cryptographic primitive is secure in the random oracle model, the randomness of the random oracle must be baked into the underlying game.

Definition 4.4. *We will denote the random oracle by $\mathcal{O} : A \to B$. Two distributions X and Y are (q, t, ϵ)-indistinguishable in the random oracle model if for any oracle algorithm $D^{\mathcal{O}}$ running in time t making at most q random oracle calls,*

$$\left| \Pr_{\mathcal{O} \xleftarrow{\$} \{f:A\to B\}}[D^{\mathcal{O}}(X) = 1] - \Pr_{\mathcal{O} \xleftarrow{\$} \{f:A\to B\}}[D^{\mathcal{O}}(Y) = 1] \right| \leq \epsilon$$

For simplicity, we will typically set $q = t$. We will define PRG security in the random oracle model to be identical to typical PRG security, but with the computational indistinguishability to be also set in the random oracle model.

Definition 4.5. *Two distributions X and Y are (s, t, ϵ)-indistinguishable in the AI-ROM if for any oracle function $z^{\mathcal{O}}$ into strings of length s and for any oracle algorithm $D^{\mathcal{O}}$ running in time t,*

$$\left| \Pr_{\mathcal{O} \xleftarrow{\$} \{f:A \to B\}} [D^{\mathcal{O}}(z^{\mathcal{O}}, X) = 1] - \Pr_{\mathcal{O} \xleftarrow{\$} \{f:A \to B\}} [D^{\mathcal{O}}(z^{\mathcal{O}}, Y) = 1] \right| \leq \epsilon$$

We similarly define PRG security in the AI-ROM.

To prove security in the AI-ROM, we can reduce security to a simpler presampling model. In the presampling model, the random oracle has a adversarially chosen preprogrammed value on p of its points, and elsewhere is uniformly random. Note that in order to achieve negligible security loss in the reduction, it is necessary that $\frac{1}{p}$ is negligible [22].

Theorem 4.6. *[41](Presampling Lemma): We say that a function f agrees with a partial assignment $\{a_1 \to b_1, \ldots, a_k \to b_k\}$ if for all $i \in [k]$, $f(a_i) = b_i$. Let p be a positive integer. Let $z^{\mathcal{O}}$ be an oracle function into strings of length s.*

Then there is an oracle function $I^{\mathcal{O}}$ outputting a partial assignment of length $\leq p$ such that for any oracle algorithm D making at most q queries to its oracle,

$$\left| \Pr_{\mathcal{O} \xleftarrow{\$} \{f:A \to B\}} [D^{\mathcal{O}}(z^{\mathcal{O}}) \to 1] - \Pr_{\substack{\mathcal{O} \xleftarrow{\$} \{f:A \to B\} \\ \mathcal{P} \xleftarrow{\$} \{f:A \to B | f \text{ agrees with } I^{\mathcal{O}}\}}} [D^{\mathcal{P}}(z^{\mathcal{O}}) \to 1] \right| \leq \sqrt{\frac{sq}{2p}}$$

Informally, this says that any primitive secure in the AI-ROM is secure in the ROM with p inputs adversarially assigned with security loss of $\sqrt{\frac{sq}{2p}}$.

Definition 4.7. *A two-input function C is a (t, q, δ, δ')-secure strong 2-immunizer in the ROM (respectively AI-ROM), if for any PRG (K, G) which is (t, q, δ) publicly secure in the ROM, the PRG $(K, C(G, G))$ is a (t, q, δ') backdoor secure PRG in the ROM (respectively AI-ROM).*

The definition of a (t, q, δ, δ')-secure weak 2-immunizer in the ROM/AI-ROM will be analogous.

Note that in particular our definition for 2-immunizer security in the AI-ROM only requires that the underlying PRG be secure in the ROM. This is a stronger definition, and we do this to model the situation where the auxiliary input represents a backdoor for the underlying PRGs.

4.2 Random Oracle is a 2-Immunizer

To show that a random oracle is a strong 2-immunizer, we adapt the proof structure from [21]. That is, we prove a key information theoretic property about publicly secure PRGs, and then use this property to bound the probability that some adversary queries the random oracle on key values.

In particular, let G^X, G^Y be two PRGs with outputs $x_1, \ldots, x_q$ and $y_1, \ldots, y_q$, and let RO be a random oracle. We will argue that the only part of the PRG game for $RO(G^X, G^Y)$ which queries $RO(x_i, y_i)$ is when the 2-immunizer is directly called by the game. This is because all parts of the game will only have access to at most one of x_i or y_i, and so therefore as the other is information theoretically unpredictable, they will be unable to query x_i and y_i to the oracle at the same time.

Afterwards, we will show that RO is still a strong 2-immunizer even in the presence of auxiliary input. We will show this by using the presampling lemma (Theorem 4.6). The trick we will use is that since our key property is information theoretic, we can set p for the presampling lemma to be exponential in λ, and so the security loss we suffer will be negligible.

We begin by stating the following information theoretic lemma. The proof will be deferred to Section 4.3.

Lemma 4.8. *(KEY LEMMA) Let $K : \{0,1\}^\ell \to \mathcal{PK} \times \mathcal{SK}$, $G : \mathcal{PK} \times \mathcal{S} \to \{0,1\}^n \times \mathcal{S}$ be a (t, q, δ) publicly secure PRG. Let $r \in \{0,1\}^\ell$ be some initial randomness. For $p \in (0,1)$, we say that r is p-weak if for $(pk, sk) \leftarrow K(r)$,*

$$\max_{\widetilde{x} \in \{0,1\}^n} \Pr_{x_1, \ldots, x_q \xleftarrow{\$} \mathbf{out}^q(G_{pk}, \mathcal{U}_\mathcal{S})} [x_i = \widetilde{x} \text{ for some } i \in [q]] \geq p.$$

Denote

$$p' := \Pr_{r \in \{0,1\}^\ell}[r \text{ is } p\text{-weak}]$$

Then,

$$p' \cdot p^2 \leq q^2 \left(\delta + \frac{q}{2^n}\right).$$

Intuitively, we call a public key pk (described using its initial randomness r) weak if the output of G_{pk} is predictable. The above lemma gives an upper bound on the probability of a public key being weak. That is, we show (through an averaging argument) that every publicly secure PRG has unpredictable output for most choices of its public parameters.

We now proceed to the proof of Theorem 4.1.

Proof. Let $K : \{0,1\}^\ell \to \mathcal{PK} \times \mathcal{SK}$, $G : \mathcal{PK} \times \mathcal{S} \to \{0,1\}^n \times \mathcal{S}$ be a (t, q, δ)-secure PRG. Let D be a distinguisher against $f(G, G)$ running in time t. Let $HONEST$ be the distribution

$$(sk, pk) \xleftarrow{\$} K, s_X, s_Y \xleftarrow{\$} \mathcal{S}$$

$$(pk, sk, \mathbf{out}^q(C(G_{pk}, G_{pk}), (s^X, s^Y)))$$

and let $RANDOM$ be the distribution

$$(sk, pk) \xleftarrow{\$} K, (r_1, \ldots, r_q) \xleftarrow{\$} \mathcal{U}_{qm}$$

$$(pk, sk, r_1, \ldots, r_q)$$

We want to bound

$$\delta' = |\Pr[D(HONEST) = 1] - \Pr[D(RANDOM) = 1]|$$

Let q_K, q_G, q_D be bounds on the number of times K, G, D query the random oracle respectively. Note that these are all bounded by t.

Let us consider the case where the distinguisher is given the output of the honest 2-immunizer. We will denote $\mathbf{out}^q(G, s^X) = x_1, \ldots, x_q$ and $\mathbf{out}^q(G, s^Y) = y_1, \ldots, y_q$. Let BAD be the event that there is some i such that (x_i, y_i) is queried to the random oracle more than once. Note that conditioned on $\overline{BAD}$, the two distributions in the distinguishing game are identical. Thus, $\delta' \leq \Pr[BAD]$.

We will break BAD up into five cases, and bound each case separately.

- We define BAD_1 to be the event where there exists i, j such that $x_i = x_j$ and $y_i = y_j$. This corresponds to (x_i, y_i) be queried to the random oracle more than once by the game itself.

- We define BAD_2 to be the event that K queries x_i, y_i for some i.

- We define BAD_3 to be the event that G queries x_i, y_i in the process of calculating $\mathbf{out}^q(G_{pk}, s^X)$.

- We define BAD_4 to be the event that G queries x_i, y_i in the process of calculating $\mathbf{out}^q(G_{pk}, s^Y)$.

- We define BAD_5 to be the event that D queries x_i, y_i.

Lemma 4.9. $\Pr[BAD_1] \leq \delta + \dfrac{q^2}{2^n}$

First, we will bound $\Pr[BAD_1]$. Let $\mathcal{A}$ be an attacker for the underlying PRG game on (K, G) which on input $r_1, \ldots, r_q$ outputs 1 if $r_i = r_j$ for some $i \neq j$. It is clear that $\Pr[\mathcal{A}(pk, \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})) \rightarrow 1] \geq \Pr[BAD_1]$, and $\Pr[A(pk, \mathcal{U}_{qn}) \rightarrow 1] \leq \dfrac{q^2}{2^n}$. But by public security of the PRG, $\Pr[\mathcal{A}(pk, \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})) \rightarrow 1] - \Pr[A(pk, \mathcal{U}_{qn}) \rightarrow 1] \leq \delta$ Thus, we have

$$\Pr[BAD_1] \leq \delta + \frac{q^2}{2^n}$$

Lemma 4.10. $\Pr[BAD_2] \leq qq_K\sqrt{\delta + \dfrac{q}{2^n}}$

We will bound $\Pr[BAD_2]$ using the key lemma. We claim that

$$\Pr_{r \xleftarrow{\$} \mathcal{U}_\ell}[r \text{ is } p\text{-weak}] \geq \sqrt{\Pr[BAD_2]}$$

for some suitable value of p. We will then use the key lemma to get an upper bound on $\Pr[BAD_2]$.

Let r be such that

$$\Pr[BAD_2 | (pk, sk) \leftarrow K(r)] \geq \sqrt{\Pr[BAD_2]}$$

We claim then that r is p-weak for some p to be specified later. Let F_r be the set of random oracle queries made by $K(r)$. We can more precisely state

$$\Pr[BAD_2 | (pk, sk) \leftarrow K(r)] = \Pr[(x_i, y_i) \in F_r \text{ for some index } i | (pk, sk) \leftarrow K(r)]$$

In particular, we can ignore one output and see that this means

$$\Pr_{x_1, \ldots, x_q \xleftarrow{\$} \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})}[x_i \in F_r \text{ for some index } i] \geq \sqrt{\Pr[BAD_2]}$$

But since $|F_r| \leq q_K$, this means there must be some element $\widetilde{x} \in F_r$ such that

$$\Pr_{x_1,\ldots,x_q \xleftarrow{\$} \mathbf{out}^q(G_{pk},\mathcal{U}_\mathcal{S})}[x_i = \widetilde{x} \text{ for some index } i] \geq \frac{\sqrt{\Pr[BAD_2]}}{q_K}.$$

But this precisely means that r is p-weak, for $p = \dfrac{\sqrt{\Pr[BAD_2]}}{q_K}$. Thus,

$$\sqrt{\Pr[BAD_2]}\Pr[BAD_2] \leq q_K^2 q^2 \left(\delta + \frac{q}{2^n}\right)$$

and so as

$$\Pr[BAD_2]^2 \leq \sqrt{\Pr[BAD_2]}\Pr[BAD_2],$$

we have

$$\Pr[BAD_2] \leq q q_K \sqrt{\delta + \frac{q}{2^n}}.$$

Lemma 4.11. $\Pr[BAD_3] \leq q^2 q_G \sqrt{\delta + \dfrac{q}{2^n}}$

To bound $\Pr[BAD_3]$, we will again use the key lemma and show that

$$\Pr_{r \xleftarrow{\$} \mathcal{U}_\ell}[r \text{ is } p\text{-weak}] \geq \sqrt{\Pr[BAD_3]}$$

for some suitable value of p.

Let r be such that

$$\Pr[BAD_3 | (pk, sk) \leftarrow K(r)] \geq \sqrt{\Pr[BAD_3]}$$

. We claim then that r is p-weak for some p to be specified later. Note that since this probability is the average over s of $\Pr[BAD_3 | (pk, sk) \leftarrow K(r), s^X = s]$, there must be some $\widetilde{s}$ such that

$$\Pr[BAD_3 | (pk, sk) \leftarrow K(r), s^X = \widetilde{s}] \geq \sqrt{\Pr[BAD_3]}.$$

Let $F_{r,\widetilde{s}}$ be the queries made by G when calculating $\mathbf{out}^q(G_{pk}, \widetilde{s})$. Using a similar argument as in the previous paragraph, we see that there must be some pair $(\widetilde{x}, \widetilde{y}) \in F_{r,\widetilde{s}}$ such that

$$\Pr_{y_1,\ldots,y_q \xleftarrow{\$} \mathbf{out}^q(G_{pk},\mathcal{U}_\mathcal{S})}[y_i = \widetilde{y} \text{ for some index } i] \geq \frac{\sqrt{\Pr[BAD_3]}}{|F_{r,\widetilde{s}}|}.$$

But note that $|F_{r,\widetilde{s}}| \leq q \cdot q_G$ as it is generated by running G q times. Thus, r is p-weak for $p = \dfrac{\sqrt{\Pr[BAD_3]}}{q \cdot q_G}$. The same algebra as the previous lemma gives us

$$\Pr[BAD_3] \leq q^2 q_G \sqrt{\delta + \frac{q}{2^n}}$$

Lemma 4.12. $\Pr[BAD_4] \leq q^2 q_G \sqrt{\delta + \dfrac{q}{2^n}}$

The proof of this lemma is analogous to the proof for $\Pr[BAD_3]$.

Lemma 4.13. $\Pr[BAD_5] \leq qq_D\sqrt{\delta + \frac{q}{2^n}}$

To bound $\Pr[BAD_5]$, we first notice that at the point when D first queries x_i, y_i, the only information available to D is the secret key and the output of $i-1$ random oracle calls. As at this point D has never queried any of its inputs, the probability that D succeeds at querying any input is the same as if D were given only the secret key.

Let us fix any initial randomness $r \in \{0,1\}^\ell$ such that

$$\Pr[BAD_5|(pk, sk) \leftarrow K(r)] \geq \sqrt{\Pr[BAD_5]}$$

. We can clearly see that

$$\Pr[BAD_5|(pk, sk) \leftarrow K(r)]$$
$$\leq \max_{\substack{F \subseteq \{0,1\}^n \\ |F| \leq q_D}} \Pr[(x_i, y_i) \in F \text{ for some index } i|(pk, sk) \leftarrow K(r)]$$

But by union bound we then have

$$\Pr[BAD_5|(pk, sk) \leftarrow K(r)] \leq q_D \max_{\widetilde{x} \in \{0,1\}^n} \Pr[x_i = \widetilde{x} \text{ for some } i \in [q]].$$

The same reasoning as the previous arguments shows us that r is p-weak for $p = \frac{\sqrt{\Pr[BAD_5]}}{q_D}$. Applying the key lemma gives us

$$\Pr[BAD_5] \leq qq_D\sqrt{\delta + \frac{q}{2^n}}$$

Putting all the lemmas together, we have

$$\delta' \leq \Pr[BAD] \leq \left(\delta + \frac{q^2}{2^n}\right) + \left(qq_K + 2q^2q_G + qq_D\right)\sqrt{\delta + \frac{q}{2^n}}$$

Noting that $q_K, q_G, q_D \leq t$ gives us our theorem. □

We then proceed to the proof of Theorem 4.3.

Proof. We will use the presampling lemma to show this. Namely, let p be some integer to be set later. Let I^{RO} be the partial assignment function of length p defined by the presampling lemma. Let $\mathcal{F} = \{f : A \to B\}$ and let $\mathcal{F}_{I^{\mathcal{O}}} = \{f : A \to B | f \text{ agrees with } I^{\mathcal{O}}\}$. We want to show

$$\left| \Pr_{\substack{\mathcal{O} \xleftarrow{\$} \mathcal{F} \\ RO \xleftarrow{\$} \mathcal{F}_{I^{\mathcal{O}}}}} [D(HONEST) \to 1] - \Pr_{\substack{\mathcal{O} \xleftarrow{\$} \mathcal{F} \\ RO \xleftarrow{\$} \mathcal{F}_{I^{\mathcal{O}}}}} [D(RANDOM) \to 1] \right| \leq \delta'$$

for some suitable δ'.

We will define BAD to be the event that there is some i such that (x_i, y_i) is queried to the random oracle more than once or is contained in $I^{\mathcal{O}}$. As before, if conditioned on BAD not occurring, $HONEST$ and $RANDOM$ are identically distributed. We will also define $BAD_1, \ldots, BAD_5$ as in the main proof. Note that the bounds for each $\Pr[BAD_i]$ determined earlier apply also in this model.

Define BAD_6 to be the event that (x_i, y_i) is contained in $I^{\mathcal{O}}$. As $|I^{\mathcal{O}}| \leq p$, the same argument as for BAD_2 gives us that

$$\Pr[BAD_6] \leq qp\sqrt{\delta + \frac{q}{2^n}}$$

Thus,

$$\delta' \leq negl(\lambda) + qp\sqrt{\delta + \frac{q}{2^n}}$$

Applying the key lemma then gives us that in the AI-ROM,

$$\mathbf{CD}_t(HONEST, RANDOM) \leq negl(\lambda) + qp\sqrt{\delta + \frac{q}{2^n}} + 2 \cdot \sqrt{\frac{st}{2p}}$$

Thus, it remains to set p such that this is negligible. Taking $p = \frac{1}{\sqrt[4]{\delta + \frac{q}{2^n}}}$ gives us

$$\mathbf{CD}_t(HONEST, RANDOM) \leq negl(\lambda) + q\sqrt[4]{\delta + \frac{q}{2^n}} + \sqrt{2st}\sqrt[8]{\delta + \frac{q}{2^n}} = negl(\lambda)$$

□

4.3 Proof of Lemma 4.8

Proof. We will define an attacker $\mathcal{A}$ against the PRG game for (K, G), and then we will bound p' using its success probability. We will define our attacker $\mathcal{A}$ as follows: On input $r_1, \ldots, r_q$, $\mathcal{A}$ will sample $s' \xleftarrow{\$} \mathcal{S}$ and calculate $r'_1, \ldots, r'_q$. If $r_i = r'_i$ for any i, $\mathcal{A}$ outputs 1. Otherwise, $\mathcal{A}$ outputs 0.

Let r be some p-weak initial randomness with $K(r) \to (pk, sk)$. By definition, there exists some $\widetilde{x}$ such that

$$\Pr_{x_1,\ldots,x_q \xleftarrow{\$} \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})}[x_i = \widetilde{x} \text{ for some } i \in [q]] \geq p.$$

Thus, there exists some index i such that

$$\Pr[x_i = \widetilde{x}] \geq \frac{p}{q}.$$

But then by independence we have

$$\Pr_{\substack{x_1,\ldots,x_q \xleftarrow{\$} \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}}) \\ y_1,\ldots,y_q \xleftarrow{\$} \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}})}}[x_i = y_i = \widetilde{x}] \geq \frac{p^2}{q^2}.$$

Thus,

$$\Pr[\mathcal{A}(pk, \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}}) \to 1 | r \text{ is } p\text{-weak}] \geq \frac{p^2}{q^2}$$

But

$$\begin{aligned}\Pr[\mathcal{A}(pk, \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}}) \to 1] &\geq p' \Pr[\mathcal{A}(pk, \mathbf{out}^q(G_{pk}, \mathcal{U}_{\mathcal{S}}) \to 1 | r \text{ is } p\text{-weak}] \\ &= p' \frac{p^2}{q^2}.\end{aligned}$$

Note that by union bound, $\Pr[\mathcal{A}(pk, \mathcal{U}_{qn}) \to 1] \leq \frac{q}{2^n}$. Thus, we have

$$p' \frac{p^2}{q^2} \leq \delta + \frac{q}{2^n}.$$

□

5 Black Box Separation (with Limitations)

Definition 5.1. *Let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a function. We call an input $x \in \{0,1\}^n$ "left-bad" if $\max_{z \in \{0,1\}^m} \Pr_{y \in \{0,1\}^n}[C(x,y) = z] > \frac{1}{2}$. We define what it means for an input to be "right-bad" analogously.*

We say that C is highly dependent on both inputs if

$$\Pr_{(x,y) \xleftarrow{\$} \{0,1\}^{2n}}[x \text{ is "left-bad" OR } y \text{ is "right-bad"}] \leq negl(\lambda).$$

Informally, a two-input function C is highly dependent on both inputs if it ignores one of its inputs at most a negligible proportion of the time. This is a rather broad category of functions. In particular, XOR, pairings, inner product, and random oracles are all highly dependent on both inputs. Furthermore, any collision resistant hash function must also be highly dependent on both inputs, otherwise it would be trivial to find a collision.

We show that it is hard to prove security (either weak or strong) for any 2-immunizer C which is highly dependent on both inputs. Note that one of the most common and useful techniques for proving security of cryptographic primitives is to create a black box reduction to some cryptographic assumption. Informally, a black box reduction transforms an attacker for some cryptographic primitive into an attacker for a cryptographic assumption. Thus, if the cryptographic assumption is immune to attack, the cryptographic primitive will be secure.

We show that if a 2-immunizer is highly dependent on both inputs, then there cannot be any black-box reduction of its security to any falsifiable cryptographic assumption.

Theorem 5.2 (Theorem 1.4 restated)**.** *Let C be a weak 2-immunizer which is highly dependent on both inputs. If there is a black-box reduction showing that C is $(poly(\lambda), \lambda, negl(\lambda), negl(\lambda))$-secure from the security of some cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

As a random oracle is highly dependent on both inputs, any reasonable hash function should also be highly dependent on both inputs. This implies that despite the fact that a random oracle is a strong 2-immunizer, it may be hard to argue security for any particular instantiation of the random oracle.

On Seeded 1-Immunizers Dodis et. al. [21] bypass the impossibility result for immunizers with public randomness in their model by considering the notion of semi-private immunizers. In a semi-private immunizer, the immunizer has access to some randomness **seed**. The randomness will be semi-private in the sense that it will be available to the distinguishing adversary, but not to the immunized PRG. This models the scenario where the designer of the PRG has no knowledge of the immunizer beforehand.

Definition 5.3. *Let $f_{\boldsymbol{seed}} : \{0,1\}^n \to \{0,1\}^m$ be a function family keyed by $\boldsymbol{seed}$. We say that a function $f_{\boldsymbol{seed}} : \{0,1\}^n \to \{0,1\}^m$ is a semi-private (t, q, δ, δ')-immunizer if for any (t, q, δ) publicly secure PRG (K, G)*

$$(pk, sk) \xleftarrow{\$} K, \boldsymbol{seed} \xleftarrow{\$} \{0,1\}^s$$

$$\boldsymbol{CD}_t((pk, sk, \boldsymbol{seed}, \boldsymbol{out}^q(f_{\boldsymbol{seed}}(G_{pk}), \mathcal{U}_{\mathcal{S}})), (pk, sk, \boldsymbol{seed}, \mathcal{U}_{qn})) \leq \delta.$$

We prove an analogue of Theorem 5.2 in this model. The proof will also follow the framework of Theorem 6.2 from [44]. Note that our separation in this model is unconditional in the sense that we make no requirements on the structure of the immunizer.

Theorem 5.4. *Let $f_{\boldsymbol{seed}} : \{0,1\}^n \to \{0,1\}^m$ be a semi-private $(poly(\lambda), q, negl(\lambda), negl(\lambda))$-immunizer such that $q \geq \lambda$ and $m \geq \lambda$. If PRGs exist and there is a black-box reduction showing that C is $(poly(\lambda), \lambda, negl(\lambda),$
$negl(\lambda))$-secure from the security of some cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

5.1 Our Black-Box Separation Techniques

The simulatable attacker paradigm The simulatable attacker paradigm, first introduced by [13] and formalized by [44], is a method for transforming a black-box reduction into an attack against the underlying assumption. This paradigm was first used to prove black-box separations from all falsifiable assumptions in [25].

In particular, let $\mathcal{C}$ be a cryptographic protocol with a black-box reduction to a cryptographic assumption $\mathcal{G}$. Formally, we will describe the black-box reduction as an oracle algorithm $\mathcal{B}^{\cdot}$ which breaks the security of $\mathcal{G}$ whenever its oracle is a (possibly inefficient) adversary breaking the security of $\mathcal{C}$.

A *simulatable attack* against $\mathcal{C}$ is an (inefficient) attack $\mathcal{A}$ which breaks the security game of $\mathcal{C}$, but which can be simulated by an efficient algorithm **Sim**. In particular, oracle access to $\mathcal{A}$ and **Sim** should be indistinguishable to the black-box reduction $\mathcal{B}^{\cdot}$. If this occurs, then since $\mathcal{B}^{\mathbf{Sim}}$ is indistinguishable from $\mathcal{B}^{\mathcal{A}}$, $\mathcal{B}^{\mathbf{Sim}}$ is an efficient attack breaking the security game of $\mathcal{G}$.

Note that in order for this paradigm to make sense, it needs to be the case that the simulator has more capabilities than the inefficient adversary, otherwise the simulator itself would be an

attack for $\mathcal{C}$. In practice, this is done by either restricting the oracle queries made by the black-box separation $\mathcal{B}$ or by restricting the power of the attacker $\mathcal{A}$.

Black-box separations for 2-stage games In 2013, Wichs showed a a general framework for proving that two-stage games cannot be reduced to any falsifiable assumption [44]. In a two-stage security game the adversary consists of two algorithms which each have individual state, but are not allowed to communicate. Thus, a simulatable attack consists of the inefficient attack as well as two simulators where the simulators *do* have shared state. This means that it is conceivable to have a simulator **Sim** for which oracle access is *indistinguishable* from $\mathcal{A}$.

Note that if we have a simulatable attack of this form, then this simulator will fool every (efficient) black-box reduction. Thus, if we can prove that for every construction there exists an simulatable attack, this gives a black-box separation of the security definition from any falsifiable assumption.

Simulatable attack against any 2-immunizer Note that an adversary against a 2-immunizer consists of both a set of PRGs and a distinguisher. Here, the PRGs and the distinguisher are not allowed to share state, and so we can hope to construct a simulatable attack in the style of [44].

Given C any candidate 2-immunizer, let G^X, G^Y be random functions and let $D(y)$ be the algorithm which outputs 1 if there exists an (s^X, s^Y) such that $y = \mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y))$. It is clear that G^X, G^Y, D is an inefficient attack breaking the security of C.

To simulate this, we simply replace G^X, G^Y with a lazy sampling oracle. That is, the first time G^X sees s, it will respond with a random value, and it will use the same response for future queries of s. To simulate D, the simulator will check if there exists *an already queried* (s^X, s^Y) such that $y = \mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y))$. Since the adversary is polynomially bounded, there will only be a polynomial number of already queried points, and so this simulator is efficient.

It turns out that the only way to distinguish this simulator from the inefficient adversary is to find some y such that $y = \mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y))$ for either s^X or s^Y unqueried. If neither s^X or s^Y has been queried before, then by a counting argument it is impossible to guess such a y. But if s^X has been queried before, if C ignores s^Y then it is possible to guess $\mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y))$ without querying s^Y. To avoid this problem, we simply assume that the output of C is dependent on both of its inputs, as in Definition 5.1.

Simulatable attack against any semi-private immunizer The idea behind this attack follows the same structure as for 2-immunizers. That is, the inefficient distinguisher will simply manually check whether the output it received is in the image space of the immunizer applied to a random function G. Then, a similar argument to the 2-immunizer case shows that this attack is simulatable.

The corresponding condition that we need to make on a semi-private immunizer $f_{\mathbf{seed}}$ is that the output of $f_{\mathbf{seed}}$ is dependent on its input x. But we can show that this is true for any good semi-private immunizer. Thus, as long as pseudorandom generators exist (and so immunizing is non-trivial), we get an unconditional black-box separation of semi-private immunizers from any falsfiable assumption.

5.2 Preliminaries

Definition 5.5. *[44, 26]A cryptographic game $\mathcal{G} = (\Gamma, c)$ is defined by a (possibly inefficient) random system Γ (the challenger) and a constant $c \in [0, 1)$. On security parameter λ, the challenger $\Gamma(1^\lambda)$ interacts with some attacker $\mathcal{A}(1^\lambda)$ and outputs a bit b. We denote this interaction by $b = (\mathcal{A}(1^\lambda) \rightleftarrows \Gamma(1^\lambda))$. The advantage of an attacker $\mathcal{A}$ in the game $\mathcal{G}$ is defined as*

$$\mathrm{Adv}_{\mathcal{G}}^{\mathcal{A}} := \Pr[(\mathcal{A}(1^\lambda) \rightleftarrows \Gamma(1^\lambda)) \to 1] - c$$

Definition 5.6. *Let $\mathcal{C}$ be some cryptographic property and let $\mathcal{G}$ by a cryptographic game. A black-box reduction deriving the security of $\mathcal{C}$ from the security of $\mathcal{G}$ is an oracle-access PPT machine $\mathcal{B}^{(\cdot)}$ for which there are some constants $c, N_0 > 0$ such that for all integers $\lambda \geq N_0$ and all (possibly inefficient, non-uniform) oracles $\mathcal{A}_n$ with $\mathrm{Adv}_{\mathcal{C}}^{\mathcal{A}_\lambda}(\lambda) \geq \frac{1}{2}$, we have $\mathrm{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}_\lambda}}(\lambda) \geq n^{-c}$.*

Definition 5.7. *A simulatable attack on a cryptographic property $\mathcal{C}$ consists of the following:*

-A finite indexing set $\mathcal{H}_\lambda$

-An ensemble of inefficient stateless non-uniform attackers $\mathcal{A}_{\lambda,h}$ indexed by $n \in \mathbb{N}$, $h \in \mathcal{H}_\lambda$

*-A stateful PPT simulator **Sim***

such that the following properties hold

- For each $\lambda \in \mathbb{N}, h \in \mathcal{H}_\lambda$, the inefficient attacker $\mathcal{A}_{\lambda,h}$ successfully breaks the security of $\mathcal{C}$ with advantage $1 - negl(\lambda)$.

- For every (possibly inefficient) oracle access machine $\mathcal{M}$ making at most $poly(\lambda)$ queries to its oracle

$$\left| \Pr_h(\mathcal{M}^{\mathcal{A}_{\lambda,h}}(1^\lambda) \to 1) - \Pr(\mathcal{M}^{\boldsymbol{Sim}(1^\lambda)}(1^\lambda) \to 1) \right| \leq negl(\lambda)$$

We remark that the initial statement of this definition in [44] requires advantage 1. Our change is purely notational. For any cryptographic primitive we can redefine advantage by adding some negligible function without affecting the security of the protocol.

Theorem 5.8. *If there exists a simulatable attack against some cryptographic notion $\mathcal{C}$ and there is a black-box reduction showing the security of $\mathcal{C}$ from the security of some cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

5.3 Black Box Separation for 2-Immunizers (Proof of Theorem 5.2)

This argument will loosely follow the framework of Theorem 6.2 from [44], showing that there is a simulatable attack against correlation-resistant one-way functions. The key difference is that

instead of relying on some degeneracy check, we use the fact that C is highly dependent on both inputs to guarantee that "bad" honest initializations will occur with negligible frequency.

The other issue we must handle with care is that the inefficient adversary must win the PRG game for all indices $h \in \mathcal{H}$, which we handle by explicitly requiring that every h turns the inefficient adversary into a PRG. We then show that almost all functions are in $\mathcal{H}$, and so the simulator can simply treat h as a random function.

Theorem 5.9. *Let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be highly dependent on both inputs and let $q \geq \lambda$.*

Then there exists an ensemble of simulatable attackers

$$\mathcal{A}_h = ((K^X, G^X), (K^Y, G^Y), D)$$

against the $(poly(\lambda), q, negl(\lambda), negl(\lambda))$-security of C as a weak 2-immunizer.

Proof. First, we will define the indexing set for our adversary. Let $h : [q] \times \{0,1\}^{\lambda/4} \to \{0,1\}^n$ be a function. Define $G_h(count, s) := (h(count, s), (count + 1, s))$. We say that h is good if G_h is a $(poly(\lambda), q, negl(\lambda))$-secure PRG. We define

$$\mathcal{H} = \{(h^X, h^Y) | h^X, h^Y : [q] \times \{0,1\}^{\lambda/4} \to \{0,1\}^n \text{ and } h^X, h^Y \text{ are both good}\}.$$

We will also state a simple lemma claiming that our indexing set makes up a large proportion of the set of all functions.

Lemma 5.10. *Let $\mathcal{F} = \{f : [q] \times \{0,1\}^{\lambda/4} \to \{0,1\}^n\}$ be the set of all functions. Then*

$$\Pr_{f \xleftarrow{\$} \mathcal{F}}[f \text{ is good}] \geq 1 - negl(\lambda).$$

The proof of this lemma is deferred to Section A.1.

Our goal is to define an $\mathcal{A}_h$ such that

$$(pk^X, sk^X) \xleftarrow{\$} K^X, (pk^Y, sk^Y) \xleftarrow{\$} K^Y, pk = (pk^X, pk^Y), sk = (sk^X, sk^Y)$$

$$HONEST = (pk, sk, \mathbf{out}^q(C(G^X, G^Y), (\mathcal{S}^X, \mathcal{S}^Y)))$$

$$RANDOM = (pk, sk, \mathcal{U}_{qm})$$

$$|\Pr[D(HONEST) = 1] - \Pr[D(RANDOM) = 1]| \geq 1 - negl(\lambda)$$

and $(K^X, G^X), (K^Y, G^Y)$ are $(poly(\lambda), q, negl(\lambda))$-secure PRGs.

We also need to define some stateful simulator **Sim** such that any inefficient $\mathcal{M}^{(\cdot)}$ making a polynomially bounded number of oracle calls cannot distinguish between oracle access to $\mathcal{A}_h$ and **Sim**.

We set $\mathcal{A}_h$ to be as defined in Figure 3.

We then must define the simulator **Sim** in Figure 4.

K^X:
Output $\perp$

K^Y:
Output $\perp$

$G^X(count, s)$:
If $count > q$, output $\perp$.
$x \leftarrow h^X(count, s)$
Output $(x, (count + 1, s))$.

$G^Y(count, s)$:
If $count > q$, output $\perp$.
$y \leftarrow h^Y(count, s)$
Output $(y, (count + 1, s))$.

$D(r_1, \ldots, r_q)$:
For each $s^X, s^Y \in \{0,1\}^{\lambda/4}$
- If $(r_1, \ldots, r_q) = \mathbf{out}^q(C(G^X, G^Y), ((0, s^X), (0, s^Y)))$, output 1
Output 0

Figure 3: Inefficient adversary for any C highly dependent on both inputs

$\mathbf{Sim}(K^X)$:
Output $\perp$

$\mathbf{Sim}(K^Y)$:
Output $\perp$

$\mathbf{Sim}(G^X, count, s)$:
If $(count, s) \in T^Y$
-Output $T^X[count][s]$
$r \xleftarrow{\$} \{0,1\}^n$
$T^X[s][count] = r$
Output r

$\mathbf{Sim}(G^Y, count, s)$:
If $(count, s) \in T^Y$
-Output $T^Y[count][s]$
$r \xleftarrow{\$} \{0,1\}^n$
$T^Y[s][count] = r$
Output r

$\mathbf{Sim}(D, r_1, \ldots, r_q)$:
For each $s^X \in T^X$, $s^Y \in T^Y$
- If $(r_1, \ldots, r_q) = \mathbf{out}^q(C(G^X, G^Y), ((0, s^X), (0, s^Y)))$, output 1.
Output 0.

Figure 4: Simulator indistinguishable from inefficient adversary. **Sim** tracks internal arrays $T^X, T^Y \in [q] \times \{0,1\}^{\lambda/4} \times \{0,1\}^n$.

Thus, all that remains is to show the following:
1. The PRGs defined by $\mathcal{A}$ win the PRG game.
2. The distinguisher defined by $\mathcal{A}$ distinguishes correctly.
3. $\mathcal{A}$ and **Sim** are indistinguishable.

Proof of 1: This is trivial by the definition of $\mathcal{H}$.

Proof of 2: Note that by construction if D is provided genuine immunized output, then it will output 1 with probability 1. Thus, it just remains to bound the probability that D outputs 1 on truly random input. Note that D compares its input to any of $|\{0,1\}^{\lambda/4}| \cdot |\{0,1\}^{\lambda/4}| = 2^{\lambda/2}$ values, and outputs 1 if and only if its input matches one of these values. However, a random value is sampled from $2^{qn} \geq 2^{\lambda}$ possible choices. Thus, the probability that D returns 1 on a uniformly random input is $\leq \frac{2^{\lambda/2}}{2^{\lambda}} = 2^{-\lambda/2}$.

Proof of 3: We will show this by a series of hybrids starting with some attacker interacting with $\mathcal{A}$, and ending with an attacker interacting with **Sim**. As a sketch of the argument, the first hybrid will involve replacing the hash functions used in $\mathcal{A}$ with truly random functions.

- Hybrid 0: Defined as $\mathcal{A}$
- Hybrid 1: Defined as Hybrid 0, but with $\mathcal{H}$ replaced with the set of all functions. By the lemma, this will be indistinguishable from the original $\mathcal{A}$ even by a computationally inefficient distinguisher.
- Hybrid 2: Defined as **Sim**. Let $\mathcal{M}$ be some (possibly inefficient) adversary distinguishing hybrids 1 and 2 in polynomially many queries. Let E be the event where $\mathcal{M}$ interacting with hybrid 1 makes a D query on input $r_1, \ldots, r_1$ such that D returns 1 and $r \neq \mathbf{out}^q(C(G^X, G^Y), ((0, s^X), (0, s^Y))$ for any s^X previously queried to G or for any s^Y previously queried to G^Y and for any value of *count*. Note that if we consider a new set of oracles which acts as hybrid 2, but has D return 0 whenever a query would cause E to be true, then the distribution of $\mathcal{M}$ interacting with this set of oracles is identical to that of $\mathcal{M}$ interacting with hybrid 1. Thus, $\mathcal{M}$ succeeds with probability $\leq \Pr[E]$.

  We define E^0 to be the event where neither s^X nor s^Y were queried to G^X or G^Y, we define E^X to be the event where s^X was queried to G^X but s^Y was not queried, and we define E^Y to be the analogue for Y. It is clear that $\Pr[E] \leq \Pr[E^0] + \Pr[E^X] + \Pr[E^Y]$. Note that $\Pr[E^0] \leq poly(\lambda) \cdot \frac{2^{\lambda/2}}{2^{\lambda}} = negl(\lambda)$ since it is the same as the probability of hitting a point in a random set of $2^{\lambda/2}$ values out of 2^{λ} possibilities using only $poly(\lambda)$ queries. Thus, it remains to bound $\Pr[E^X]$ (the case for $\Pr[E^Y]$ will be analogous).

  Let R be the event that for some s^X queried, one of $x_1, \ldots, x_q$
  $= \mathbf{out}^q(G^X, s^X)$ is "left-bad". Since each x_i is chosen uniformly at random and as C is highly dependent on both inputs, by the union bound $\Pr[R] \leq negl(\lambda) \cdot q$·number of queries made to D. But q and the number of queries made to D are both polynomial, so $\Pr[R] \leq negl(\lambda)$.

  We will now condition on R not occurring, and show that $\Pr[E^X|\overline{R}] \leq negl(\lambda)$. Let $E^X_{i,j}$ be the event that E^X is triggered during the ith query to D and the corresponding s^X was queried in the jth query to G^X. We will denote by $r_1, \ldots, r_q$ the value queried in the ith

query to D and we will denote $x_1, \ldots, x_q = \mathbf{out}^q(G^X, s^X)$.

$$\begin{aligned}
&\Pr[E^X_{i,j}|\overline{R}] \\
&\leq \Pr_{h^Y}\left[\exists \text{ unqueried } s^Y \text{ s.t. } \mathbf{out}^q(C(G^X, G^Y), (s^X, s^Y)) = r_1, \ldots, r_q|\overline{R}\right] \\
&\leq 2^{\lambda/4} \Pr_{y_1,\ldots,y_q \xleftarrow{\$} \mathcal{U}_{qm}}[r_i = C(x_i, y_i) \text{ for all } i \in [q]|\overline{R}] \qquad (3) \\
&\leq 2^{\lambda/4}\frac{1}{2^q} \leq 2^{\lambda/4-\lambda} = 2^{-\frac{3}{4}\lambda} = negl(\lambda)
\end{aligned}$$

But as G^X and D can only be queried a polynomial number of times, by the union bound $\Pr[E^X|\overline{R}] \leq negl(\lambda)$. So we have

$$\Pr[E^X] \leq \Pr[E^X \cap \overline{R}] + \Pr[E^X \cap R] \leq \Pr[E^X|\overline{R}] + \Pr[R] \leq negl(\lambda)$$

Therefore, we have $\Pr[E] \leq negl(\lambda)$ and so hybrids 1 and 2 are computationally indistinguishable.

□

Combining Theorem 5.9 and Theorem 5.8 gives us a proof of Theorem 5.2.

5.4 Black Box Separation for Semi-private Immunizers (Proof of Theorem 5.4)

The proof of Theorem 5.4 follows the same framework as the proof of Theorem 5.2. That is, we will prove that there exists a simulatable attack against any semi-private immunizer, and then use Theorem 5.8 to prove the main result. More formally, we prove the following theorem:

Theorem 5.11. *Let $f_{\boldsymbol{seed}} : \{0,1\}^n \to \{0,1\}^m$ be a semi-private $(poly(\lambda), q, negl(\lambda), negl(\lambda))$-immunizer such that $q \geq \lambda$ and $m \geq \lambda$. If there exists a $(poly(\lambda), q, negl(\lambda))$-publicly secure PRG $\widetilde{G}$, then there exists an ensemble of simulatable attackers*

$$\mathcal{A}_h = ((K, G), D)$$

against the $(poly(\lambda), q, negl(\lambda))$-security of $f_{\boldsymbol{seed}}$.

Proof. We say that $h : [q] \times \{0,1\}^{\lambda/2} \to \{0,1\}^n$ is good if it is good in the sense of Lemma 5.10. We define $\mathcal{H} = \{[q] \times \{0,1\}^{\lambda/4} \to \{0,1\}^n : h \text{ is good}\}$

Our goal is to define an $\mathcal{A}_h$ such that

$$(pk, sk) \xleftarrow{\$} K$$

$$HONEST = (pk, sk, \mathbf{seed}, \mathbf{out}^q(f_{\mathbf{seed}}(G), \mathcal{S}))$$

$$RANDOM = (pk, sk, \mathbf{seed}, \mathcal{U}_{qm})$$

$$|\Pr[D(HONEST) = 1] - \Pr[D(RANDOM) = 1]| \geq 1 - negl(\lambda)$$

and (K, G) is a $(poly(\lambda), q, negl(\lambda))$-secure PRGs.

We also need to define some stateful simulator **Sim** such that any inefficient $\mathcal{M}^{(\cdot)}$ making a polynomially bounded number of oracle calls cannot distinguish between oracle access to $\mathcal{A}_h$ and **Sim**.

Note that our algorithm will perform a test to ensure that the **seed** it is given is not degenerate. For our purposes, we will call a **seed** degenerate if there exists a y such that $\Pr_x[f_{\mathbf{seed}}(x) = y] \geq \frac{1}{2}$.

K:
Output $\perp$

$G(count, s)$:
If $count > q$, output $\perp$.
$x \leftarrow h^X(count, s)$
Output $(x, (count + 1, s))$.

$D(\mathbf{seed}, r_1, \ldots, r_q)$:
Degeneracy check: sample q random values $x_1, \ldots, x_q$. If $f_{\mathbf{seed}}(x_i) = f_{\mathbf{seed}}(x_j)$ for some i, j, output 0.
For each $s \in \{0,1\}^{\lambda/4}$
- If $(r_1, \ldots, r_q) = \mathbf{out}^q(f_{\mathbf{seed}}(G), s)$, output 1
Output 0

Figure 5: Inefficient adversary for any $f_{\mathbf{seed}}$

We set $\mathcal{A}_h$ to be as defined in Figure 5.

We then must define the simulator **Sim** in Figure 6.

Thus, all that remains is to show the following:
1. The PRGs defined by $\mathcal{A}$ win the PRG game.
2. The distinguisher defined by $\mathcal{A}$ distinguishes correctly.
3. $\mathcal{A}$ and **Sim** are indistinguishable.

Proof of 1: This is trivial by the definition of $\mathcal{H}$ and by Lemma 5.10.

Proof of 2: Set $\epsilon = \Pr_{\mathbf{seed}, x_1, \ldots, x_q}[f_{\mathbf{seed}}(x_i) \neq f_{\mathbf{seed}}(x_j) \text{ for all } i, j]$. We have that the probability that D outputs 1 on an immunized input is exactly ϵ. Note that by construction, if D is provided genuine immunized output, then it will output 1 with probability $1 - \epsilon$. But we must have $\epsilon \leq negl(\lambda)$ as otherwise we can distinguish $\mathbf{out}^q(f_{\mathbf{seed}}(\widetilde{G}), s)$ from random by simply checking if there are any repetitions.
Thus, it just remains to bound the probability that D outputs 1 on truly random input. Note that D compares its input to any of $|\{0,1\}^{\lambda/2}| = 2^{\lambda/2}$ values, and outputs 1 if and only if its input

Sim(K):
Output $\perp$

Sim$(G, s, count)$:
If $(s, count) \notin T$
-Output $T[s][count] \xleftarrow{\$} \{0,1\}^n$
Output $T[count][s]$

Sim$(D, \mathbf{seed}, r_1, \ldots, r_q)$:
Degeneracy check: we check if **seed** is degenerate using the same degeneracy check as the inefficient adversary. That is, we sample $x_1, \ldots, x_q$ and if $f_{\mathbf{seed}}(x_i) = f_{\mathbf{seed}}(x_j)$ for some i, j, output 0.
For each $s \in T$
- If $(r_1, \ldots, r_q) = \mathbf{out}^q(f_{\mathbf{seed}}(G), s))$, output 1.
Output 0.

Figure 6: Simulator indistinguishable from inefficient adversary. **Sim** tracks an internal array $T \in (\{0,1\}^{\lambda/4} \times [q]) \times \{0,1\}^n$.

matches one of these values. However, a random value is sampled from $2^{qn} \geq 2^\lambda$ possible choices. Thus, the probability that D returns 1 on a uniformly random input is $\leq \frac{2^{\lambda/2}}{2^\lambda} = 2^{-\lambda/2}$.
Therefore, $\mathcal{A}$ breaks $f_{\mathbf{seed}}$ with advantage $(1 - negl(\lambda)) - negl(\lambda) = 1 - negl(\lambda)$.

Proof of 3: We will show this by a series of hybrids starting with some attacker interacting with $\mathcal{A}$, and ending with an attacker interacting with **Sim**. As a sketch of the argument, the first hybrid will involve replacing the hash functions used in $\mathcal{A}$ with truly random functions.

- Hybrid 0: Defined as $\mathcal{A}$
- Hybrid 1: Defined as Hybrid 0, but with $\mathcal{H}$ replaced with the set of all functions. By the lemma, this will be indistinguishable from the original $\mathcal{A}$ even by a computationally inefficient distinguisher.
- Hybrid 2: Defined as **Sim**. Let $\mathcal{M}$ be some (possibly inefficient) adversary distinguishing hybrids 1 and 2 in polynomially many queries. Let E be the event where $\mathcal{M}$ interacting with hybrid 1 makes a D query on input $\mathbf{seed}, r_1, \ldots, r_1$ such that D returns 1 and $r \neq \mathbf{out}^q(C(G), s)$ for any s previously queried to G for any value of *count*. Note that if we consider a new set of oracles which acts as hybrid 2, but has D return 0 whenever a query would cause E to be true, then the distribution of $\mathcal{M}$ interacting with this set of oracles is identical to that of $\mathcal{M}$ interacting with hybrid 1. Thus, $\mathcal{M}$ succeeds with probability $\leq \Pr[E]$.
  Let E_i be the event that E is triggered during the ith query to D. Let R_i be the event that

the **seed** queried in round i is degenerate. Then

$$\Pr[E_i \cap R_i] \leq \Pr[E_i|R_i] \leq \Pr[\textbf{seed} \text{ passes the degeneracy check}] \leq negl(q) = negl(\lambda)$$

It remains to bound $\Pr[E_i \cap \overline{R_i}]$.

$$\begin{aligned} &\Pr[E_i|\overline{R_i}] \\ &\leq \Pr_h\left[\exists \text{ unqueried } s \text{ s.t. } \mathbf{out}^q(C(G), s) = r_1, \ldots, r_q|\overline{R_i}\right] \\ &\leq 2^{\lambda/2} \Pr_{x_1,\ldots,x_q}[r_i = f_{\textbf{seed}}(x_i) \text{ for all } i \in [q]|\overline{R_i}] \\ &\leq 2^{\lambda/2}\frac{1}{2^q} \leq \frac{1}{2^{\lambda/2}} = negl(\lambda) \end{aligned} \tag{4}$$

But as G and D can only be queried a polynomial number of times, by the union bound $\Pr[E] \leq negl(\lambda)$. Thus, hybrids 1 and 2 are computationally indistinguishable.

□

References

[1] Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology: Special Publication, 2012. `http://csrc.nist.gov/publications/PubsSPs.html#800-90A`.

[2] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[3] Michael Alekhnovich. More on average case vs approximation complexity. *computational complexity*, 20(4):755–786, Dec 2011.

[4] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `https://eprint.iacr.org/2005/385`.

[5] Giuseppe Ateniese, Danilo Francati, Bernardo Magri, and Daniele Venturi. Immunization against complete subversion without random oracles. *Theor. Comput. Sci.*, 859:1–36, 2021.

[6] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `https://eprint.iacr.org/2005/417`.

[7] Balthazar Bauer, Pooya Farshim, and Sogol Mazaheri. Combiners for backdoored random oracles. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 272–302. Springer, Heidelberg, August 2018.

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

[9] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, August 2014.

[10] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[11] Rishiraj Bhattacharyya, Mridul Nandi, and Anik Raychaudhuri. Crooked indifferentiability of enveloped XOR revisited. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, volume 13143 of *Lecture Notes in Computer Science*, pages 73–92. Springer, 2021.

[12] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

[13] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998.

[14] Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *SIGACT News*, 41(3):59–85, sep 2010.

[15] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual ec incident. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 468–479, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 319–335. USENIX Association, August 2014.

[17] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In *26th FOCS*, pages 429–442. IEEE Computer Society Press, October 1985.

[18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Heidelberg, April / May 2018.

[19] Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless Fruit is the sweetest: Random number generation, revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 205–234. Springer, Heidelberg, August 2019.

[20] Yevgeniy Dodis, Pooya Farshim, Sogol Mazaheri, and Stefano Tessaro. Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity. In Rafael Pass and

Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 241–273. Springer, Heidelberg, November 2020.

[21] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, April 2015.

[22] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 473–495. Springer, Heidelberg, April / May 2017.

[23] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372. Springer, Heidelberg, August 2016.

[24] Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. Extracting randomness from extractor-dependent sources. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 313–342. Springer, Heidelberg, May 2020.

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

[26] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 202–219. Springer, Heidelberg, March 2009.

[27] Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 77–92. Springer, Heidelberg, August 2002.

[28] Thibaut Horel, Sunoo Park, Silas Richelson, and Vinod Vaikuntanathan. How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 42:1–42:20. LIPIcs, January 2019.

[29] Ari Juels and Jorge Guajardo. RSA key generation with verifiable randomness. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 357–374. Springer, Heidelberg, February 2002.

[30] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, April 2015.

[31] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[32] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: Private communication against a dictator. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 34–63. Springer, Heidelberg, May / June 2022.

[33] Willy Quach, Brent Waters, and Daniel Wichs. Targeted lossy functions and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 424–453, Virtual Event, August 2021. Springer, Heidelberg.

[34] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. Cryptology ePrint Archive, Report 2015/695, 2015. `https://eprint.iacr.org/2015/695`.

[35] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, December 2016.

[36] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. Cryptology ePrint Archive, Paper 2016/530, 2016. `https://eprint.iacr.org/2016/530`.

[37] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 907–922. ACM Press, October / November 2017.

[38] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Correcting subverted random oracles. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 241–271. Springer, Heidelberg, August 2018.

[39] Dan Shumow and Niels Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng. In *Proc. Crypto'07*, 2007. `https://rump2007.cr.yp.to/15-shumow.pdf`.

[40] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[41] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Heidelberg, August 2007.

[42] Umesh V. Vazirani and Vijay V. Vazirani. Trapdoor pseudo-random number generators, with applications to protocol design. In *24th FOCS*, pages 23–30. IEEE Computer Society Press, November 1983.

[43] Umesh V. Vazirani and Vijay V. Vazirani. Efficient and secure pseudo-random number generation. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 193–202. Springer, Heidelberg, August 1984.

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

[45] Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, August 1996.

[46] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.

[47] Adam Young and Moti Yung. Kleptography from standard assumptions and applications. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 271–290. Springer, Heidelberg, September 2010.

A Supplemental Materials

A.1 Proof of Lemma 5.10

Proof. Let $p(\lambda)$ be some polynomial. Let $\mathcal{F} = \{f : [q] \times \{0,1\}^{\lambda/4} \to \{0,1\}^n\}$. We will use the probabilistic method to argue that a large fraction of functions $f \in \mathcal{F}$ are such that if we define $G_f(count, s) = (f(count, s), (count + 1, s))$, then $\mathbf{CD}_{p(\lambda)}(\mathbf{out}^q(G_f(\mathcal{U}_\mathcal{S}), \mathcal{U})) \leq negl(\lambda)$.

Let C be any cicuit of size $p(\lambda)$. Define $p_C = \Pr[C(\mathcal{U}) \to 1]$. We want to bound

$$\Pr_{f \xleftarrow{\$} \mathcal{F}}[|p_C - \Pr[C(\mathbf{out}^q(G_f(\mathcal{U}_\mathcal{S}))) \to 1]| \geq \delta]$$

for some δ to be set later.

We will define random variables $\widetilde{p_C} = \Pr[C(\mathbf{out}^q(G_f(\mathcal{U}_\mathcal{S}))) \to 1]$ and $T_s = C(\mathbf{out}^q(G_f(s)))$. Then,

$$\widetilde{p_C} = \frac{1}{2^{\lambda/4}} \sum_{s \in \{0,1\}^{\lambda/4}} T_s.$$

But note that as f is uniformly random, for any single given s $G_f(s)$ is uniformly random. Thus, $\mathbb{E}[T_s] = p_C$.

By the Chernoff bound,

$$\Pr[|p_C - \widetilde{p_C}| \geq \delta] \leq 2e^{-\frac{\delta^2 2^{\lambda/4}}{3p}}$$

Thus, taking $\delta = \frac{1}{2^{\lambda/16}}$ gives

$$\Pr_{f \xleftarrow{\$} \mathcal{F}}[|p_C - \widetilde{p_C}| \geq \frac{1}{2^{\lambda/16}}] \leq 2e^{-\frac{2^{\lambda/8}}{3}}.$$

But note that there are $2^{O(p(\lambda)\log(p(\lambda)))}$ circuits of size $p(\lambda)$. Thus, by the union bound,

$$\Pr_{f \xleftarrow{\$} \mathcal{F}}[\text{There exists } C \text{ such that } |p_C - \widetilde{p_C}| \geq \frac{1}{2^{\lambda/16}}] \leq 2^{O(p(\lambda)\log(p(\lambda)))} 2e^{-\frac{2^{\lambda/8}}{3}}$$

But $2^{O(p(\lambda)\log(p(\lambda)))} 2e^{-\frac{2^{\lambda/8}}{3}} = negl(\lambda)$ and so we are done. □