






# Immunizing Backdoored PRGs

Marshall Ball<sup>(✉)</sup>, Yevgeniy Dodis<sup>(iD)</sup>, and Eli Goldin<sup>(iD)</sup>

New York University, New York, USA  
{marshall,dodis}@cs.nyu.edu, eli.goldin@nyu.edu

**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. [22] 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  $k = 1$ , [22] 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 [8] gave supporting evidence that simple  $k$ -immunizers might exist, albeit in slightly different settings. Unfortunately, we show that simple standard model proposals of [8, 37] (including the XOR function [8]) 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

---

M. Ball—Supported in part by the Simons Foundation.

Y. Dodis—Research Supported by NSF grant CNS-2055578, and gifts from JP Morgan, Protocol Labs and Algorand Foundation.

E. Goldin—Partially supported by a National Science Foundation Graduate Research Fellowship.

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 Vazirani [42, 43] introduced the notion of “trapdoored PRGs” and showed the Blum-Blum-Shub PRG is one such example [13]. 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 [17, 22, 39] 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,<sup>1</sup> 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 [17, 39] showed that the above attack idea can be extended to attacking the actual NIST PRG. Moreover, the famous “Juniper Dual EC incident” (see [16] 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. [22] 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  $sk$ .

<sup>1</sup> And instead thinking of PRG as outputting pseudorandom elliptic curve points.

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  $X = (R_1, \dots, R_q)$  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, [22] 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 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 [5, 35]. 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 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*.<sup>2</sup>

The question of building such immunizers was formally introduced and studied by Dodis et al. [22]. 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

<sup>2</sup> 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.

$Z_i = C(R_i)$  instead of  $R_i$  itself. The hope being that hashing the output of a PRG will provide security even against the suspected backdoor  $sk$ .<sup>3</sup> Unfortunately, [22] 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 [22] 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 [22] 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) [9]. Thus, we ask the question:

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

It turns out that we can use the elegant black-box separation technique of Wichs [44] to give a negative answer to this question (proof included in the full version [6]).

**Theorem 1.** *If there is a black-box reduction showing security of a seeded immunizer  $C$  from the security of some cryptographic game  $\mathcal{G}$ , then  $\mathcal{G}$  is not secure.*

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 [22] 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 [22] 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.<sup>5</sup> We call such post-processing procedures *2-immunizers*.<sup>6</sup> More generally, one can consider  $k$ -immunizers for  $k \geq 2$ , but

<sup>3</sup> This assumption presumes that such  $C$  itself is not backdoored.

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

<sup>5</sup> 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.

<sup>6</sup> Drawing inspiration from 2-source extractors [18] to similarly overcome the impossibility of deterministic extraction from a single weak source of randomness.

setting  $k = 2$  is obviously the most preferable in practice. As before, our hope would be that the final outputs  $(Z_1, \dots, Z_q)$  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 et al. [8, 21] and Russell et al. [37].

DETOUR 1: BACKDOORED RANDOM ORACLES. In this model [8], 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 (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, [8] 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 [15] for a survey of this problem). Thus, despite not settling this question unconditionally, the results of [8] 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 [45–47] (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 [8] 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.<sup>7</sup> 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 [22] (and this work) allows the attacker to build  $F$  and  $G$  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 (Enc, Dec) which has pseudorandom ciphertexts, and is what we call *C-homomorphic*. Oversimplifying for the sake of presentation (see Definition 13), we need an encryption scheme where the message

<sup>7</sup> Under a widely believed cryptographic assumption mentioned shortly.

$m$ —independently encrypted twice under the same public key  $pk$  with corresponding ciphertexts  $x$  and  $y$ —can still be recovered using the secret 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 2.** *Assuming the Alekhnovich assumption (listed in Proposition 1) 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 [36,37] 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.<sup>8</sup> 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 [36,37] in Remark 3.2 is too strong, and no construction can meet it.<sup>9</sup>

To see this modeling inadequacy directly, recall that one of the standard model  $k$ -immunizers from [36,37] 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

<sup>8</sup> 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.

<sup>9</sup> 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.

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  $i$ th bit of Dual-EC initialized with  $s$ . Syntactically, this is the same (very dangerous) 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 [36,37] to our setting, we can still ask if the random oracle 2-immunizer result claimed by [36,37] is actually true for backdoored PRGs. Fortunately, we show that this is indeed the case, by giving a direct security proof.<sup>10</sup> In fact, it works even in the so called *auxiliary-input ROM* (AI-ROM) defined by Unruh [41] and recently studied by [19,23]. 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 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  $x$  and  $y$* , 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 18, but intuitively states that there are few “influential” inputs  $x^*$  (resp.,  $y^*$ ) which

<sup>10</sup> In particular, the key piece of our proof that was missing in [36,37], is contained in Lemma 8 of our paper. The important observation (adapted from the seeded 1-immunizers proof in [22]) 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.



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 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.*

**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 [4, 7].

**Theorem 5.** *Assuming the SXDH assumption (listed in Conjecture 1) holds for groups  $G_X, G_Y, G_T$ , a bilinear map  $e : G_X \times G_Y \rightarrow G_T$  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, [22] built seeded 1-immunizers based on the existence of so called universal computational extractors (UCEs) [9]. 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 [18], these results can either be overcome using seeded extractors [31], or two-source extractors [18].

A special class of seeded extractors 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* [25,33], and generalize the corresponding notion of oracle-dependent extractors considered by [20] in the ROM. Conceptually similar to our results, [25] 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 [20].

KLEPTOGRAPHY. Young and Yung studied what they called kleptography: subversion of cryptosystems by modifying encryption algorithms in order to leak information subliminally [45–47]. 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 [10] treats a special case of Young and Yung’s setting for symmetric encryption.

As described in detail above, the works [36,37] 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 [5,34] also consider immunizing corrupted PRGs, however these results succeed 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 [27, 40]) 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 steganographic channels, even if the dictator demands the users to reveal their purported secret keys.

Finally, the works of [24, 30] 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 [8] and [12] 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 Sect. 3.

## 2 Definitions

**Definition 1.** *Two distributions  $X$  and  $Y$  are called  $(t, \epsilon)$ -indistinguishable (denoted by  $\mathbf{CD}_t(X, Y) \leq \epsilon$ ) if for any algorithm  $D$  running in time  $t$ ,*

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

**Definition 2.** *Let  $X_\lambda$  and  $Y_\lambda$  be two families of distributions indexed by  $\lambda$ . If for all polynomial  $t(\lambda)$  and some negligible  $\epsilon(\lambda)$ ,  $X_\lambda$  and  $Y_\lambda$  are  $(t(\lambda), \epsilon(\lambda))$ -indistinguishable, then we say  $X$  and  $Y$  are computationally indistinguishable (denoted by  $\mathbf{CD}(X, Y) \leq \text{negl}(\lambda)$ ).*

### 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 [22]:

**Definition 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

- $K : \{0, 1\}^\ell \rightarrow \mathcal{PK} \times \mathcal{SK}$  takes in randomness and outputs a public key  $pk$  and secret key  $sk$ . We will denote running  $K$  on uniform input as  $(pk, sk) \stackrel{\$}{\leftarrow} K$ .
- $G : \mathcal{PK} \times \mathcal{S} \rightarrow \{0, 1\}^n \times \mathcal{S}$  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 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  $(r_1, \dots, r_q)$  the output of  $(K, G)$ , and denote it by  $\mathbf{out}^q(G_{pk}, s)$  (or  $\mathbf{out}^q(G, s)$ ).

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

**Definition 5.** A PRG  $(K, G)$  is a  $(t, q, \delta)$  publicly secure PRG if  $K, G$  both run in time  $t$  and

$$pk \stackrel{\$}{\leftarrow} K$$

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

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 6.** A PRG  $(K, G)$  is a  $(t, q, \delta)$  backdoor secure PRG if  $K, G$  both run in time  $t$  and

$$(pk, sk) \stackrel{\$}{\leftarrow} K$$

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

Note that there are PRGs that are  $(t, q, \delta)$  publicly secure, but not  $(t', q, \delta')$  backdoor secure even for some  $t' \ll t$  and  $\delta' \gg \delta$  [22]. 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 [22]. Note in particular that while the [22] 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 7.** Let  $(K^X, G^X), (K^Y, G^Y)$  be two PRGs and let  $C : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^m$  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  $K^X \rightarrow pk^X, sk^X$ ,  $K^Y \rightarrow pk^Y, sk^Y$  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  $\mathbf{out}^q(G^X, s^X) = x_1, \dots, x_q$  and  $\mathbf{out}^q(G^Y, s^Y) = y_1, \dots, y_q$ , then

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

**Definition 8.** A two-input function  $C$  is a  $(t, q, \delta, \delta')$ -secure weak 2-immunizer, if for any  $(t, q, \delta)$  publicly secure PRGs  $(K^X, G^X), (K^Y, G^Y)$ , the PRG  $((K^X, K^Y), C(G^X, G^Y))$  is a  $(t, q, \delta')$  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 9.** Let  $(K, G)$  be a PRG and let  $C : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^m$  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  $x_1, \dots, x_q = \mathbf{out}^q(G_{pk}, s^X)$  and  $y_1, \dots, y_q = \mathbf{out}^q(G_{pk}, s^Y)$  are two outputs of  $G$  on the same public key and freshly sampled initial states, then

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

**Definition 10.** A two-input function  $C$  is called a  $(t, q, \delta, \delta')$ -secure strong 2-immunizer, if for any  $(t, q, \delta)$  publicly secure PRG  $(K, G)$ , the PRG  $(K, C(G, G))$  is a  $(t, q, \delta')$  backdoor secure PRG.

**Lemma 1.** *If  $C$  is a  $(t, q, \delta, \delta')$ -secure strong 2-immunizer, then  $C$  is a  $(t, q, \delta, 4\delta')$ -secure weak 2-immunizer.*

For a proof of this lemma, see a full version of this paper [6].

*Remark 1.* Some traditional definitions of PRGs [11] 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 6 (Theorem 2 restated).** *Assuming the Alekhovich assumption (listed in Proposition 1) holds, XOR is not a  $(\text{poly}(\lambda), 1, \text{negl}(\lambda), \text{negl}(\lambda))$ -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 11.** Let  $G_X, G_Y, G_T$  be groups of prime order exponential in  $\lambda$  with generators  $g_X, g_Y, g_T$ . A bilinear map  $e : G_X \times G_Y \rightarrow G_T$  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 7 (Theorem 5 restated).** Assuming the SXDH assumption (listed in Conjecture 1) holds for groups  $G_X, G_Y, G_T$ , a bilinear map  $e : G_X \times G_Y \rightarrow G_T$  is not a  $(poly(\lambda), 2, negl(\lambda), negl(\lambda))$ -secure weak 2-immunizer.

Note that although [8] 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 (Gen, Enc, 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 12.** We say that a public key encryption scheme (Gen, Enc, Dec) is pseudorandom if for all  $m$ ,

$$pk \xleftarrow{\$} \text{Gen}$$

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

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

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

**Theorem 8.** *Let  $(\text{Gen}, \text{Enc}, \text{Dec})$  be a public key encryption scheme and let  $C^e$  be some operation. Then, if  $(\text{Gen}, \text{Enc}, \text{Dec})$  is pseudorandom and  $C^e$ -homomorphic (with homomorphic decryption algorithm  $\text{Dec}^{C^e}$ ), then  $C^e$  is not a  $(\text{poly}(\lambda), 1, \text{negl}(\lambda), \text{negl}(\lambda))$ -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  $C^e(G, G)$  has a backdoor.

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

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  $(\text{poly}(\lambda), 1, \text{negl}(\lambda))$  publicly secure PRG by the definition of a pseudorandom PKE. Thus, it remains to show an adversary  $D$  that can distinguish

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

from

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

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

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

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

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

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

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

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 Alekhovich assumption, it is Conjecture 4.7 in his paper). Instead of presenting his underlying assumption directly, we will refer to the following proposition:

**Proposition 1** [3]: *Suppose that the Alekhovich assumption holds, then for every  $m = O(n)$ ,  $k = \Theta(\sqrt{n})$ ,  $\ell, t \leq \text{poly}(n)$  then*

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

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



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 6.

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 8, XOR is not a strong 2-immunizer.

First, we present Alekhnovich’s public key encryption scheme in Fig. 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

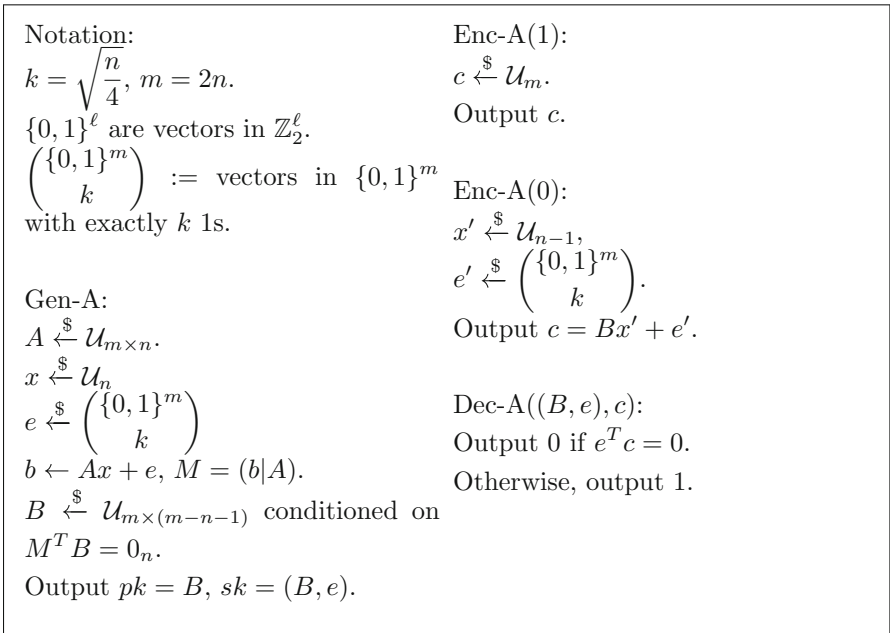


Fig. 1. Alekhnovich’s PKE scheme (From Sect. 4.4.3).

**Proposition 2** [3]: *Assuming the Alekhnovich assumption holds,*

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

**Corollary 1.** *Assuming the Alekhnovich assumption holds, (Gen-A, Enc-A, Dec-A) is pseudorandom.*

**Proposition 3.** *Assuming the Alekhovich assumption holds, (Gen-A, Enc-A, Dec-A) as presented above is  $\oplus$ -homomorphic.*

The proof of Proposition 3 is in the full version of this paper [6].

### 3.3 Weak 2-Immunizers

**Definition 14.** *Let  $C^e : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^m$  be some operation. We say a pair of public key encryption schemes (Gen, Enc, Dec) and (Gen', Enc', Dec') are jointly  $C^e$ -homomorphic if there exists some function  $\text{Dec}_{sk,sk'}^{C^e}$  such that for all  $m$ ,*

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

**Theorem 9.** *Let (Gen, Enc, Dec), (Gen', Enc', Dec') be two public key encryption schemes and let  $C^e$  be some operation. Then, if (Gen, Enc, Dec), (Gen', Enc', Dec') are pseudorandom and jointly  $C^e$ -homomorphic (with homomorphic decryption algorithm  $\text{Dec}^{C^e}$ ), then  $C^e$  is not a  $(\text{poly}(\lambda), 1, \text{negl}(\lambda), \text{negl}(\lambda))$ -secure weak 2-immunizer.*

*Proof.* This proof is analogous to the proof of Theorem 8. The corresponding PRGs are  $(K^X, G^X) = (\text{Gen}, \text{Enc}(0; s))$  and  $(K^Y, G^Y) = (\text{Gen}', \text{Enc}'(0; s))$ . The distinguisher again runs  $\text{Dec}^{C^e} \rightarrow 0$  and returns 1 if and only if  $m = 0$ .

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

We remark that the Alekhovich 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 [4, 7].

**Conjecture 1.** *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  $e : G_X \times G_Y \rightarrow G_T$ , -for uniformly random  $a, b, c \xleftarrow{\$} \mathbb{Z}_{|G_X|}$   $\mathbf{CD}((g_X^a, g_X^b, g_X^{ab}), (g_X^a, g_X^b, g_X^c)) \leq \text{negl}(\lambda)$  (the Diffie Hellman assumption holds for  $G_X$ ), -for uniformly random  $a, b, c \xleftarrow{\$} \mathbb{Z}_{|G_Y|}$   $\mathbf{CD}((g_Y^a, g_Y^b, g_Y^{ab}), (g_Y^a, g_Y^b, g_Y^c)) \leq \text{negl}(\lambda)$  (the Diffie Hellman assumption holds for  $G_Y$ ).*

Note that, as stated in Definition 11 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  $\lambda$ .

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-immunizer.

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

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

**Lemma 2.** *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 and the proof of Theorem 7 using Lemma 2 to the full version [6].

## 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 10.** *Let  $RO : \{0, 1\}^{2n} \rightarrow \{0, 1\}^m$  be a random oracle. For  $t$  sufficiently large to allow for simple computations,  $f(X, Y) = RO(X||Y)$  is a  $(t, q, \delta, \delta')$ -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 3.**  *$f(X, Y) = RO(X||Y)$  is a  $(poly(\lambda), poly(\lambda), negl(\lambda), negl(\lambda))$ -secure strong 2-immunizer in the ROM.*

**Theorem 11 (Theorem 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 10 is as follows. Even given the secret and public keys for a PRG, public security guarantees that the output of each PRG is unpredictable. Let  $x_1, \dots, x_q$  and  $y_1, \dots, y_q$  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 11 comes from standard presampling techniques [23,41], with a full proof included in the full version [6].

### 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 15.** We will denote the random oracle by  $\mathcal{O} : A \rightarrow B$ . Two distributions  $X$  and  $Y$  are  $(q, t, \epsilon)$ -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,

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

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 16.** Two distributions  $X$  and  $Y$  are  $(s, t, \epsilon)$ -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  $t$ ,

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

We similarly define PRG security in the AI-ROM.

**Definition 17.** A two-input function  $C$  is a  $(t, q, \delta, \delta')$ -secure strong 2-immunizer in the ROM (respectively AI-ROM), if for any PRG  $(K, G)$  which is  $(t, q, \delta)$  publicly secure in the ROM, the PRG  $(K, C(G, G))$  is a  $(t, q, \delta')$  backdoor secure PRG in the ROM (respectively AI-ROM).

The definition of a  $(t, q, \delta, \delta')$ -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 [22]. 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  $x_1, \dots, x_q$  and  $y_1, \dots, y_q$ , 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 ??). 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  $\lambda$ , and so the security loss we suffer will be negligible.

We begin by stating the following information theoretic lemma. The proof is in the full version of this paper [6].

**Lemma 3. (KEY LEMMA)** *Let  $K : \{0, 1\}^\ell \rightarrow \mathcal{PK} \times \mathcal{SK}$ ,  $G : \mathcal{PK} \times \mathcal{S} \rightarrow \{0, 1\}^n \times \mathcal{S}$  be a  $(t, q, \delta)$  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  $(pk, sk) \leftarrow K(r)$ ,*

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

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 10.

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

$$\begin{aligned} & (sk, pk) \xleftarrow{\$} K, s_X, s_Y \xleftarrow{\$} \mathcal{S} \\ & (pk, sk, \text{out}^q(C(G_{pk}, G_{pk}), (s^X, s^Y))) \end{aligned}$$

and let  $RANDOM$  be the distribution

$$(sk, pk) \stackrel{\$}{\leftarrow} K, (r_1, \dots, r_q) \stackrel{\$}{\leftarrow} \mathcal{U}_{qm}$$

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

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  $\mathbf{out}^q(G, s^X) = x_1, \dots, x_q$  and  $\mathbf{out}^q(G, s^Y) = y_1, \dots, y_q$ . 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.**  $\Pr[BAD_1] \leq \delta + \frac{q^2}{2^n}$

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

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

**Lemma 5.**  $\Pr[BAD_2] \leq qq_K \sqrt{\delta + \frac{q}{2^n}}$

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

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

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

$$\Pr_{x_1, \dots, x_q \stackrel{s}{\leftarrow} \text{out}^q(G_{pk}, \mathcal{U}_S)} [x_i \in F_r \text{ for some index } i] \geq \sqrt{\Pr[BAD_2]}$$

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

$$\Pr_{x_1, \dots, x_q \stackrel{s}{\leftarrow} \text{out}^q(G_{pk}, \mathcal{U}_S)} [x_i = \tilde{x} \text{ for some index } i] \geq \frac{\sqrt{\Pr[BAD_2]}}{q_K}.$$

But this precisely means that  $r$  is  $p$ -weak, for  $p = \frac{\sqrt{\Pr[BAD_2]}}{q_K}$ . 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 6.**  $\Pr[BAD_3] \leq q^2 q_G \sqrt{\delta + \frac{q}{2^n}}$

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

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

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  $\tilde{s}$  such that

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

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

$$\Pr_{y_1, \dots, y_q \leftarrow \mathbb{S}}^{\mathbb{S}} \left[ y_i = \tilde{y} \text{ for some index } i \right] \geq \frac{\sqrt{\Pr[BAD_3]}}{|F_{r,\tilde{s}}|}.$$

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

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

**Lemma 7.**  $\Pr[BAD_4] \leq q^2 q_G \sqrt{\delta + \frac{q}{2^n}}$

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

**Lemma 8.**  $\Pr[BAD_5] \leq qq_D \sqrt{\delta + \frac{q}{2^n}}$

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

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

But by union bound we then have

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

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

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

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

## 5 Black Box Separation (with Limitations)

**Definition 18.** Let  $C : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^m$  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

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

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 12 (Theorem 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  $(\text{poly}(\lambda), \lambda, \text{negl}(\lambda), \text{negl}(\lambda))$ -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. We sketch the proof of this theorem in the next subsection. For the full argument, see the full version [6].

## 5.1 Proof Sketch for Theorem 12

**THE SIMULATABLE ATTACKER PARADIGM.** The simulatable attacker paradigm, first introduced by [14] 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 [26].

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  $\mathcal{B}$  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  $\mathcal{B}$ . 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  $\mathcal{C}$ .

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 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 a simulatable attack, this gives a black-box separation of the security definition from any falsifiable assumption.

**OUR SIMULATABLE ATTACK.** 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 18.

## References

1. Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology: Special Publication (2012). <https://csrc.nist.gov/publications/PubsSPs.html#800-90A>
2. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS, pp. 298–307. IEEE Computer Society Press (2003)
3. Alekhnovich, M.: More on average case vs approximation complexity. *Comput. Complex.* **20**(4), 755–786 (2011)
4. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. *Cryptology ePrint Archive*, Report 2005/385 (2005). <https://eprint.iacr.org/2005/385>
5. Ateniese, G., Francati, D., Magri, B., Venturi, D.: Immunization against complete subversion without random oracles. *Theor. Comput. Sci.* **859**, 1–36 (2021)
6. Ball, M., Dodis, Y., Goldin, E.: Immunizing backdoored prgs. *eprint* (2023). <https://eprint.iacr.org>
7. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. *Cryptology ePrint Archive*, Report 2005/417 (2005). <https://eprint.iacr.org/2005/417>
8. Bauer, B., Farshim, P., Mazaheri, S.: Combiners for backdoored random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 272–302. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-96881-0\\_10](https://doi.org/10.1007/978-3-319-96881-0_10)
9. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40084-1\\_23](https://doi.org/10.1007/978-3-642-40084-1_23)
10. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_1](https://doi.org/10.1007/978-3-662-44371-2_1)
11. Bellare, M., Yee, B.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36563-X\\_1](https://doi.org/10.1007/3-540-36563-X_1)
12. Bhattacharyya, R., Nandi, M., Raychaudhuri, A.: Crooked indifferenciability of enveloped XOR revisited. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) INDOCRYPT 2021. LNCS, vol. 13143, pp. 73–92. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92518-5\\_4](https://doi.org/10.1007/978-3-030-92518-5_4)
13. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. *SIAM J. Comput.* **15**(2), 364–383 (1986)

14. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054117>
15. Chattopadhyay, A., Pitassi, T.: The story of set disjointness. SIGACT News **41**(3), 59–85 (2010)
16. Checkoway, S., et al.: A systematic analysis of the juniper dual ec incident. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 468–479. Association for Computing Machinery, New York (2016)
17. Checkoway, S., et al.: On the practical exploitability of dual EC in TLS implementations. In: Fu, K., Jung, J. (eds.) USENIX Security 2014, pp. 319–335. USENIX Association (2014)
18. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In: 26th FOCS, pp. 429–442. IEEE Computer Society Press (1985)
19. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78381-9\\_9](https://doi.org/10.1007/978-3-319-78381-9_9)
20. Coretti, S., Dodis, Y., Karthikeyan, H., Tessaro, S.: Seedless fruit is the sweetest: random number generation, revisited. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 205–234. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26948-7\\_8](https://doi.org/10.1007/978-3-030-26948-7_8)
21. Dodis, Y., Farshim, P., Mazaheri, S., Tessaro, S.: Towards defeating backdoored random oracles: indistinguishability with bounded adaptivity. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12552, pp. 241–273. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64381-2\\_9](https://doi.org/10.1007/978-3-030-64381-2_9)
22. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_5](https://doi.org/10.1007/978-3-662-46800-5_5)
23. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56614-6\\_16](https://doi.org/10.1007/978-3-319-56614-6_16)
24. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls—secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 341–372. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53018-4\\_13](https://doi.org/10.1007/978-3-662-53018-4_13)
25. Dodis, Y., Vaikuntanathan, V., Wichs, D.: Extracting randomness from extractor-dependent sources. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 313–342. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-45721-1\\_12](https://doi.org/10.1007/978-3-030-45721-1_12)
26. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC, pp. 99–108. ACM Press (2011)
27. Hopper, N.J., Langford, J., von Ahn, L.: Provably secure steganography. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 77–92. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45708-9\\_6](https://doi.org/10.1007/3-540-45708-9_6)
28. Horel, T., Park, S., Richelson, S., Vaikuntanathan, V.: How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In: Blum, A. (ed.) ITCS 2019, vol. 124, pp. 42:1–42:20. LIPIcs (2019)

29. Juels, A., Guajardo, J.: RSA key generation with verifiable randomness. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 357–374. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45664-3\\_26](https://doi.org/10.1007/3-540-45664-3_26)
30. Mironov, I., Stephens-Davidowitz, N.: Cryptographic Reverse Firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46803-6\\_22](https://doi.org/10.1007/978-3-662-46803-6_22)
31. Nisan, N., Zuckerman, D.: Randomness is linear in space. *J. Comput. Syst. Sci.* **52**(1), 43–52 (1996)
32. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: private communication against a dictator. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022. LNCS, vol. 13276, pp. 34–63. Springer, Heidelberg (2022). [https://doi.org/10.1007/978-3-031-07085-3\\_2](https://doi.org/10.1007/978-3-031-07085-3_2)
33. Quach, W., Waters, B., Wichs, D.: Targeted lossy functions and applications. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 424–453. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-84259-8\\_15](https://doi.org/10.1007/978-3-030-84259-8_15)
34. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: clipping the power of kleptographic attacks. *Cryptology ePrint Archive, Report 2015/695* (2015). <https://eprint.iacr.org/2015/695>
35. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Cliptography: clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53890-6\\_2](https://doi.org/10.1007/978-3-662-53890-6_2)
36. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. *Cryptology ePrint Archive, Paper 2016/530* (2016). <https://eprint.iacr.org/2016/530>
37. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 907–922. ACM Press (2017)
38. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Correcting subverted random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 241–271. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-96881-0\\_9](https://doi.org/10.1007/978-3-319-96881-0_9)
39. Shumow, D., Ferguson, N.: On the possibility of a back door in the nist sp800-90 dual ec prng. In: *Proceedings of Crypto 2007* (2007). <https://rump2007.cr.yip.to/15-shumow.pdf>
40. Simmons, G.J.: The prisoners’ problem and the subliminal channel. In: Chaum, D. (ed.) CRYPTO 1983, Plenum Press, New York, USA, pp. 51–67 (1983)
41. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74143-5\\_12](https://doi.org/10.1007/978-3-540-74143-5_12)
42. Vazirani, U.V., Vazirani, V.V.: Trapdoor pseudo-random number generators, with applications to protocol design. In: 24th FOCS, pp. 23–30. IEEE Computer Society Press (1983)
43. Vazirani, U.V., Vazirani, V.V.: Efficient and secure pseudo-random number generation (extended abstract). In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 193–202. Springer, Heidelberg (1985). [https://doi.org/10.1007/3-540-39568-7\\_17](https://doi.org/10.1007/3-540-39568-7_17)
44. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Kleinberg, R.D. (ed.) ITCS 2013, pp. 111–126. ACM (2013)
45. Young, A., Yung, M.: The dark side of “Black-Box” cryptography or: should we trust capstone? In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-68697-5\\_8](https://doi.org/10.1007/3-540-68697-5_8)

46. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997). [https://doi.org/10.1007/3-540-69053-0\\_6](https://doi.org/10.1007/3-540-69053-0_6)
47. Young, A., Yung, M.: Kleptography from standard assumptions and applications. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 271–290. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15317-4\\_18](https://doi.org/10.1007/978-3-642-15317-4_18)