A Formal Treatment of Backdoored Pseudorandom Generators

Yevgeniy Dodis[1(✉)], Chaya Ganesh[1], Alexander Golovnev[1], Ari Juels[2], and Thomas Ristenpart[3]

[1] Department of Computer Science, New York University, New York, USA
dodis@cs.nyu.edu, {chaya.ganesh,alexgolovnev}@gmail.com
[2] Jacobs Institute, Cornell Tech, New York, USA
juels@cornell.edu
[3] Department of Computer Sciences, University of Wisconsin, Madison, USA
tomrist@gmail.com

Abstract. We provide a formal treatment of backdoored pseudorandom generators (PRGs). Here a saboteur chooses a PRG instance for which she knows a trapdoor that allows prediction of future (and possibly past) generator outputs. This topic was formally studied by Vazirani and Vazirani, but only in a limited form and not in the context of subverting cryptographic protocols. The latter has become increasingly important due to revelations about NIST's backdoored Dual EC PRG and new results about its practical exploitability using a trapdoor.

We show that backdoored PRGs are equivalent to public-key encryption schemes with pseudorandom ciphertexts. We use this equivalence to build backdoored PRGs that avoid a well known drawback of the Dual EC PRG, namely biases in outputs that an attacker can exploit without the trapdoor. Our results also yield a number of new constructions and an explanatory framework for why there are no reported observations in the wild of backdoored PRGs using only symmetric primitives.

We also investigate folklore suggestions for countermeasures to backdoored PRGs, which we call *immunizers*. We show that simply hashing PRG outputs is not an effective immunizer against an attacker that knows the hash function in use. Salting the hash, however, does yield a secure immunizer, a fact we prove using a surprisingly subtle proof in the random oracle model. We also give a proof in the standard model under the assumption that the hash function is a universal computational extractor (a recent notion introduced by Bellare, Tung, and Keelveedhi).

1 Introduction

Pseudorandom number generators (PRGs) stretch a short, uniform bit string to a larger sequence of pseudorandom bits. Beyond being a foundational primitive in cryptography, they are used widely in practice within applications requiring relatively large amounts of cryptographic randomness. Seed the PRG via the

E. Oswald and M. Fischlin (Eds.): EUROCRYPT 2015, Part I, LNCS 9056, pp. 101–126, 2015.
DOI: 10.1007/978-3-662-46800-5_5

output of some (more expensive to use) source of randomness, such as a system random number generator, and then use it to efficiently generate effectively unbounded number of pseudorandom bits for the application. Unfortunately, an adversary that can distinguish such bits from uniform or, worse yet, outright predict the outputs of a PRG, almost invariably compromises security of higher level applications. This fragility in the face of poor pseudorandom sources is borne out by a long history of vulnerabilities [7,8,14,16,17,22,24,33].

Perhaps it is no coincidence, then, that PRGs have also been a target for backdoors. As far back as 1983, Vazirani and Vazirani [30,31] introduce the notion of trapdoored PRGs and show the Blum-Blum-Shub PRG is one [10]. Their purpose was not for sabotaging systems, however, but instead they used the property constructively in a higher level protocol. The best known example of potential sabotage is the backdoored NIST Dual EC PRG [23]. It is parameterized by two elliptic curve points; call them P and Q. The entity that selects these points can trivially know $d = \text{dlog}_Q P$, and armed with d any attacker can from an output of the PRG predict all future outputs. This algorithm and the proposed use of it as a way of performing key escrow was detailed at length in a patent by Brown and Vanstone [11]. The possibility of the Dual EC PRG having been standardized so as to include a backdoor was first discussed publicly by Shumow and Ferguson [27]. More recent are allegations that the United States government did in fact retain trapdoor information for the P and Q constants mandated by the NIST standard. The practical implications of this backdoor, should those constants be available, were recently explored experimentally by Checkoway et al. [13]: they quantified how saboteurs might decrypt TLS sessions using the trapdoor information and sufficient computational resources.

Given the importance of backdoored PRGs (and protecting against them), we find it striking that there has been, thus far, no formal treatment of the topic of maliciously backdoored PRGs. We rectify this, giving appropriate notions for backdoored PRGs (building off of [30]) that not only capture Dual EC, but allow us to explore other possible avenues by which a backdoored PRG might be designed, the relationships between this primitive and others, and the efficacy of potential countermeasures against backdoors. We provide an overview of each set of contributions in turn.

Backdoored PRGs. We focus on families of PRGs, meaning that one assumes a parameter generation algorithm that outputs a public set of parameters that we will call, for reasons that will become clear shortly, a public key. A generation algorithm takes a public key, the current state of the generator, and yields a (hopefully) pseudorandom output, as well as a new state. This is standard. A backdoored PRG, on the other hand, has a parameter generation algorithm that additionally outputs a trapdoor value that we will also call a secret key. A backdoored PRG should provide, to any party that has just the public key, a sequence of bits that are indistinguishable from random. To a party with the secret key these bits may be easily distinguishable or, better yet from the attacker's perspective, predictable with some reasonable success probability.

As an example, the generation algorithm for backdoored Dual EC picks a fixed group element Q, a random exponent d, and outputs as public key the pair $P = Q^d$ and Q. The secret key is d. (We use multiplicative notation for simplicity.) Generation works by taking as input an initial, random state s, and then computing $s' = P^s$ as the next state. An output is computed as all but the last 16 bits of $Q^{s'}$. (We ignore here for simplicity the possible use of additional input.) An attacker that knows d can guess the unknown 16 bits of $Q^{s'}$ and compute $P^{s'}$, the value which defines the next output as well as all future states. In practice applications allow the attacker to check guesses against future PRG outputs, allowing exact discovery of the future state (c.f., [13]).

The Dual EC PRG does not provide outputs that are provably indistinguishable from random bits, and in fact some analysis has shown that despite dropping the low 16 bits, abusable biases remain [25]. A natural question is whether one can build a similarly backdoored PRG but which is provably secure as a PRG?

We answer this in the positive. To do so, we first show a more general result: the equivalence of pseudorandom public-key encryption (PKE) and backdoored PRGs whose outputs are pseudorandom (to those without the trapdoor). Pseudorandom PKE schemes have ciphertexts that are indistinguishable from random bits. Constructions from elliptic curves include Möller's [21] and the more recent Elligator proposal [9] and its variants [2,29]. Another approach to achieve pseudorandom bits is via public-key steganography [3,12,32,36]. We give a black-box construction of backdoored PRGs from any pseudorandom PKE. To complete the equivalence we show how any secure backdoored PRG can be used to build pseudorandom PKE scheme. The latter requires using an amplification result due to Holenstein [18].

We also show how a saboteur can get by with key encapsulation mechanisms that have pseudorandom ciphertexts (which are simpler than regular PKE). A KEM encapsulate algorithm takes as input randomness and a public key, and outputs a ciphertext and a one-time-use secret key. We use this algorithm directly as a generator for a backdoored PRG: the ciphertext is the output and the session key is the next state. The secret key for decapsulation reveals the next state. Seen in this light, the Dual EC PRG is, modulo the bit truncations, an instantiation of our generic KEM construction using the ElGamal KEM.

The types of backdoored PRGs discussed thus far only allow use of a trapdoor to predict future states. We formalize another type of backdoored PRG which requires the attacker to be able to determine any output (as chosen at random from a sequence of outputs) using another output (again chosen at random from the same sequence). Such "random access" could be useful to attackers that want to predict previous outputs from future ones, for example.

Immunization countermeasures. So far we have formalized the problem and discussed improved backdoored PRGs. We now turn to countermeasures, a topic of interest given the reduced trust in PRGs engendered by the possibility of backdooring. While the best countermeasure would be to use only trusted PRGs, this may not be possible in all circumstances. 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. Another oft-suggested route, therefore, is to efficiently post-process the output of a PRG in order to prevent exploitation of the backdoor. We call such a post-processing strategy an *immunizer*.

A clear candidate for an immunizer is a cryptographic hash function, such as SHA-256 (or SHA-3). A natural assumption is that hashing the output of a PRG will provide security even when the attacker knows the trapdoor, as the hash will hide the data the attacker might use to break PRG security. (This assumption presumes that SHA-256 is itself not backdoored; we have no evidence otherwise, although see [1].) Another, similar idea is to truncate a large number of the output bits.

We show that successful immunization is, perhaps surprisingly, more subtle than naïve approaches like this would suggest. We show that, *a saboteur that knows the immunizer strategy ahead of time can build a backdoored PRG that bypasses the immunizer*. We refer to this setting as the *public immunizer* security model, as both the PRG designer and the backdoor exploiter know the exact immunizer function. We show that for *any* such immunizer, the attacker can leak secret state bit-by-bit. Hence, this is true even when hashing and truncating, and even when modeling the hash function as a random oracle (RO).

This observation suggests that a the designer of a secure PRG should not have exact knowledge of the immunizer. We introduce two further security models for immunizers. In the *semi-private* model, the immunizer can use randomness unknown to the PRG designer, but which is revealed to the backdoor exploiter. In the *private* model, the randomness is never revealed to the saboteur. Constructing provably strong immunizers is straightforward in this last model, but not necessarily practical ones.

For semi-private immunizers, one can prevent basic immunizer-bypassing attacks against hashing (such as we describe below) by using the immunizer's randomness as a salt for the hash. While this immunization strategy thwarts such attacks, proving that it is secure — meaning that its outputs are provably indistinguishable from random bits even for an attacker that has the trapdoor secret of the original backdoored RNG *and the immunization salt* — is surprisingly tricky. One would like to argue that since the PRG must be indistinguishable from random to attackers without the secret trapdoor, then they must have high entropy, and hence hashing with a salt can extract uniform bits from these unpredictable outputs. However, the distinguisher here *does* know the trapdoor, and thus we cannot directly use the assumed backdoored PRG's security against distinguishers who *do not* know the trapdoor. Giving an analysis in the RO model (ROM), we overcome this hurdle by exploiting the fact that, to achieve standard PRG security (no trapdoor available) across multiple invocations with fresh seeds, the backdoored PRG must have low collision probability of outputs. We can in turn use the collision probability as a bound on the predictability of outputs by an adversary, and thereby prove the security of the hashed outputs. We also extend this result to work in the standard model assuming only that the hash function is a universal computational extractor (UCE) [4].

Further related work. As already mentioned, Vazirani and Vazirani [30,31] introduce the notion of trapdoored generators and use them constructively in protocol design. We build on their notion, but require stronger security in normal operation (indistinguishability from random bits). We also generalize to other trapdoor exploitation models, and study broader connections and countermeasures. Their trapdoor PRG using Blum-Blum-Shub can be recast to work as a backdoored PRG using our KEM-style framework (the generated parity bits being the next state and the final squaring of the seed being the generator output). This approach does produce an unbounded number of bits, however, as no further bits can be produced once the final squaring is output.

Young and Yung studied what they called kleptography: subversion of cryptosystems by modifying encryption algorithms in order to leak information subliminally [34–36]. Juels and Guajardo [20] 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 [5] treats a special case of Young and Yung's setting for symmetric encryption. We treat a different case, that of PRGs, that has not yet been extensively treated (but our general setting is the same).

Goh et al. [15] investigate how to modify TLS or SSH implementations in order to leak session keys to network attackers that know a trapdoor. One could use a backdoored PRG to accomplish this; indeed this was seemingly the intent behind use of Dual EC in TLS [13]. However, their work does not try to subvert PRGs.

Some of our results, in particular the backdoored PRG that foils public immunizers, use channels that can be viewed as subliminal in the sense introduced by Simmons [28]. Our technique is also reminiscent of the one used to build secret-key steganography [19].

2 Models and Definitions

Notation. We denote the set of all binary strings of length n by $\{0,1\}^n$, and the set of all binary strings $\{0,1\}^* = \cup_{i=0}^{\infty}\{0,1\}^i$. We denote the concatenation of two bit strings s_1 and s_2 by $s_1\|s_2$. We use lsb and lsb_2 to mean the last bit and the last two bits of a bit string, respectively. We denote by $R^{\gg 1}$ and $R^{\gg 2}$ the bit strings obtained by one and two right shifts of R, respectively.

An algorithm is a function mapping inputs from some domain to outputs in some range. For non-empty sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, we denote the composition of algorithms $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ and $\mathrm{G}\colon \mathcal{Y} \to \mathcal{Z}$ by $\mathrm{F} \circ \mathrm{G}$, i.e. $(\mathrm{F} \circ \mathrm{G})(s) = \mathrm{F}(\mathrm{G}(s))$. A randomized algorithm is an algorithm with a designated bit-string input (always the last) called the coins. We write $\mathrm{F}(x; r)$ to denote the output resulting from running F on input x and coins r. We write $y \leftarrow \mathrm{F}(x; r)$ to assign y that value. We will write $\mathrm{F}(x)$ when the coins are understood from context and write $y \leftarrow\!\!\$\ \mathrm{F}(x)$ to denote picking a fresh r appropriately and running $\mathrm{F}(x; r)$. We assume r is always of some sufficient length that we leave implicit. For brevity, we often introduce algorithms without their domain and range when these are clear from context.

The running time of an algorithm is the worst-case number of steps to compute it in an abstract model of computation with unit basic operation costs. In most cases the implementation will be clear, and we will clarify when not; our results extend in straightforward ways to finer-grained models of computation.

We write $x \leftarrow\!\!\$\ \mathcal{X}$ to denote sampling a value x uniformly from a set $\mathcal{X}$. We use the same notation for non-uniform distributions, and in such cases specify the distribution. We let $\mathcal{U}_n$ denote the uniform distribution over $\{0,1\}^n$ and $\mathcal{U}_n^q$ the uniform distribution over $\mathcal{U}_n \times \cdots \times \mathcal{U}_n$ (q repeats of $\mathcal{U}_n$). For ease of notation, we abbreviate $\mathcal{U}_n$ to $\mathcal{U}$ when the length n is clear from context. Applying an algorithm (or other function) to a distribution, e.g., $\mathrm{F}(x;\mathcal{U})$, denotes the implied distribution over outputs.

PRFs, PRPs, and Encryption. We recall a number of standard cryptographic primitives.

Definition 1 (Computational Indistinguishability). *Two distributions X and Y are called* (t,ε)-computationally indistinguishable *(denoted by $\mathbf{CD}_t(X,Y) \le \varepsilon$) if for any algorithm* D *running in time t, $|\Pr[\mathrm{D}(X)=1] - \Pr[\mathrm{D}(Y)=1]| \le \varepsilon$.*

Definition 2 (Pseudorandom Function). *A family of algorithms $\{\mathrm{F}_{sk}\colon \{0,1\}^m \to \{0,1\}^n \mid sk \in \{0,1\}^k\}$ is called a* family of (t,q,δ)-pseudorandom functions *if $\mathsf{Adv}^{\mathrm{PRF}}_{\mathrm{F}} \triangleq \max_{\mathrm{D}} \mathsf{Adv}^{\mathrm{PRF}}_{\mathrm{F}}(\mathrm{D}) \triangleq \max_{\mathrm{D}}(2\,|\Pr[\mathcal{G}^{\mathrm{PRF}}_{\mathrm{F}}(\mathrm{D}) \Rightarrow \mathsf{true}] - \frac{1}{2}|) \le \delta$ where the maximum is taken over all algorithms* D *running in time t and making up to q queries to the oracle $\mathcal{O}$ (the game $\mathcal{G}^{\mathrm{PRF}}_{\mathrm{F}}(\mathrm{D})$ is shown in Fig. 1). Function $\mathcal{F}$ in Fig. 1 is a uniformly selected random function $\mathcal{F}\colon \{0,1\}^m \to \{0,1\}^n$.*

Definition 3 (Pseudorandom Permutation). *A family of functions $\{f_{\mathsf{seed}}\colon \{0,1\}^n \to \{0,1\}^n \mid \mathsf{seed} \in \{0,1\}^\ell\}$ is called a* (t,q,ε)-pseudorandom permutation *if it is a (t,q,ε)-pseudorandom function and f_{seed} is a permutation for every $\mathsf{seed} \in \{0,1\}^\ell$.*

Conventional public-key encryption (PKE) schemes meet semantic security style notions, meaning no partial information about plaintexts is leaked. Many traditional ones additionaly are such that ciphertexts are indistinguishable from uniformly chosen group elements (e.g., ElGamal). We use something slightly different still: public-key encryption (PKE) with pseudorandom ciphertexts. These schemes have ciphertexts that are indistinguishable from random *bit strings* (of appropriate length). Both theoretical and practical constructions of such public key encryption schemes were shown in [3,12,32]. Constructions from elliptic curves include [21] and [9].

Definition 4 (IND$-CPA Public Key Encryption). *A triple $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$, where $\mathrm{K} \to \{0,1\}^p \times \{0,1\}^k, pk \in \{0,1\}^p, \mathrm{Enc}_{pk}\colon \{0,1\}^m \times \{0,1\}^\rho \to \{0,1\}^n, sk \in \{0,1\}^k, \mathrm{Dec}_{sk}\colon \{0,1\}^n \to \{0,1\}^m$ is called a $(t,q,\delta)-$ IND$-CPA public key encryption scheme if*

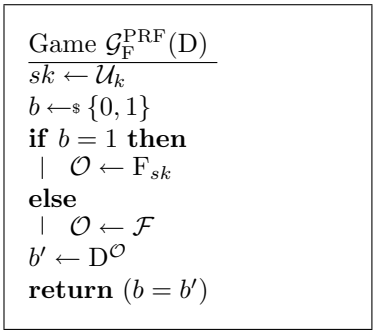

Fig. 1. PRF game

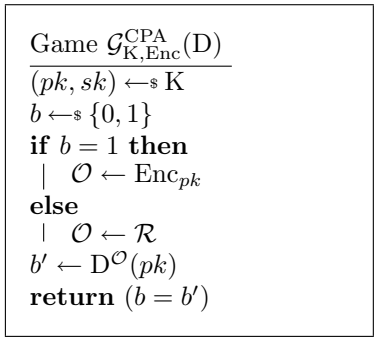

Fig. 2. IND$-CPA Game

- $\Pr[\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(s;\alpha)) = s] = 1$, *where* $s \leftarrow \{0,1\}^m, (pk, sk) \leftarrow \mathrm{K}, \alpha \leftarrow \{0,1\}^\rho$,
- $\mathsf{Adv}^{\mathrm{CPA}}_{\mathrm{K,Enc}}(\mathrm{D}) \triangleq 2\,|\Pr[\mathcal{G}^{\mathrm{CPA}}_{\mathrm{K,Enc}}(\mathrm{D}) \Rightarrow \mathsf{true}] - \frac{1}{2}| \leq \delta$ *for any algorithm* D *running in time* t *and making up to* q *queries to the oracle* $\mathcal{O}$. *(The game* $\mathcal{G}^{\mathrm{CPA}}_{\mathrm{K,Enc}}(\mathrm{D})$ *is defined in Fig. 2, the function* $\mathcal{R}$ *outputs a uniformly selected output of length* n.*)*

Pseudorandom generators. A pseudorandom generator (PRG) is a pair of algorithms (K, G). The parameter generation algorithm K takes input coins and outputs a pair (pk, sk), called the public key and secret or private key (or trapdoor). Traditionally, a PRG has no trapdoor, and pk would be referred to as the public parameter. Our notation of public / private keys is for consistency with the next section; for an ordinary PRG, sk may be taken as null. We assume that sk uniquely determines pk. A public key pk designates a family of algorithms denoted by G. Each algorithm $\mathrm{G}_{pk} \colon \mathcal{S} \to \{0,1\}^n \times \mathcal{S}$ maps an input called the state to an n-bit output and a new state. We drop the subscript pk where it is clear from context. We refer to $\mathcal{S}$ as the state space; it will often simply be bit strings of some length. We will always specify a distribution over $\mathcal{S}$ that specifies the selection of an initial state, written $s \leftarrow\!\!\$\ \mathcal{S}$, for the PRG. For any integer $q \geq 1$, we let $\mathsf{out}^q(\mathrm{G}, s)$ for $s \in \mathcal{S}$ denote the sequence of bit strings $(r_1, r_2, \ldots, r_q)$ output by running $(r_1, s_1) \leftarrow \mathrm{G}(s)$, then $(r_2, s_2) \leftarrow \mathrm{G}(s_1)$, and so on. By $\mathsf{state}^q(\mathrm{G}, s)$ we denote the sequence of states $(s_1, s_2, \ldots, s_q)$. A PRG is secure when no adversary can distinguish between its outputs and random bits.

Definition 5 (PRG security). *A PRG* (K, G) *is a* (t, q, δ)-secure PRG *if for* $pk \leftarrow \mathrm{K}$, $\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U})), \mathcal{U}) \leq \delta$.

This definition does not capture forward-secrecy, meaning that past outputs should be indistinguishable from random bits even if the current state is revealed. In all the PRG constructions that follow, we point out which of the results satisfy the forward-security notion and which are forward-insecure.

3 Backdoored Pseudorandom Generators

A backdoored pseudorandom generator (BPRG) is a triple of algorithms (K, G, A). The pair (K, G) is a PRG, as per the definition in the last section. The third algorithm A we call the adversary, although it is in fact co-designed with the rest of the scheme. It uses the trapdoor output by K to violate security of the PRG in one of several potential ways. We give games defining these distinct ways of violating security in Figure 3.

Game $\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K},\mathrm{G},\mathrm{A})$	Game $\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K},\mathrm{G},\mathrm{A})$	Game $\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{rseek}}(\mathrm{K},\mathrm{G},\mathrm{A},i,j)$
$(pk, sk) \leftarrow\!\!\$\ \mathrm{K}$	$(pk, sk) \leftarrow\!\!\$\ \mathrm{K}$	$(pk, sk) \leftarrow\!\!\$\ \mathrm{K}$
$s \leftarrow\!\!\$\ \mathcal{S}$	$s \leftarrow\!\!\$\ \mathcal{S}$	$s \leftarrow\!\!\$\ \mathcal{S}$
$r_1^0, \ldots, r_q^0 \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$	$r_1, \ldots, r_q \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$	$r_1, \ldots, r_q \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$
$r_1^1, \ldots, r_q^1 \leftarrow\!\!\$\ \mathcal{U}_n^q$	$s_1, \ldots, s_q \leftarrow\!\!\$\ \mathsf{state}^q(\mathrm{G}_{pk}, s)$	$r'_j \leftarrow\!\!\$\ \mathrm{A}(sk, i, j, r_i)$
$b \leftarrow\!\!\$\ \{0,1\}$	$s'_q \leftarrow\!\!\$\ \mathrm{A}(sk, r_1, \ldots, r_q)$	**return** $(r_j = r'_j)$
$b' \leftarrow \mathrm{A}(sk, r_1^b, \ldots, r_q^b)$	**return** $(s'_q = s_q)$	
return $(b = b')$		

Fig. 3. Security games defining success of trapdoor-equipped adversaries

The first game is identical to the standard PRG definition except that the adversary here gets the trapdoor. The second tasks A with recovering the current state, given the trapdoor and a sequence of outputs. This is, by definition, sufficient information to produce all future outputs of G_{pk}. The last tasks A with predicting the full output of some state j given the trapdoor and the output for i.

Definition 6 (Backdoored PRG). *A triple* $(\mathrm{K}, \mathrm{G}, \mathrm{A})$ *is called a* $(t, q, \delta, (\mathcal{G}_{\mathsf{type}}, \epsilon))$-backdoored PRG *for* $\mathsf{type} \in \{\mathsf{dist}, \mathsf{next}, \mathsf{rseek}\}$ *if* (K, G) *is a* (t, q, δ)*-secure PRG and* $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{type}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \geq \varepsilon$*, where*

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \triangleq 2 \cdot \left| \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \Rightarrow \mathsf{true}] - \frac{1}{2} \right|,$$

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \triangleq \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \Rightarrow \mathsf{true}], \text{ and}$$

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{rseek}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \triangleq \min_{1 \leq i,j \leq q} \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{rseek}}(\mathrm{K}, \mathrm{G}, \mathrm{A}, i, j) \Rightarrow \mathsf{true}].$$

A $\mathcal{G}_{\mathsf{dist}}$-BPRG is only interesting when $\varepsilon \gg \delta$, as otherwise the distinguisher without the trapdoor information can distinguish already with advantage δ. For the other types, even if $\varepsilon < \delta$ the definition is still meaningful.

A $(t, q, \delta, (\mathcal{G}_{\mathsf{next}}, \epsilon)$-BPRG is (strictly) better for the saboteur than achieving a $\mathcal{G}_{\mathsf{dist}}$-BPRG under the same parameters. The random seek notion is orthogonal; it may or may not be better depending on the situation. Our attacks and

(looking ahead to later sections) defenses will be given according to the strongest definitions. That is when taking on the role of the sabotuer, we will build $\mathcal{G}_{\mathsf{next}}$-BPRGs and/or $\mathcal{G}_{\mathsf{rseek}}$-BPRGs with as efficient as possible A. When considering defenses against saboteurs by way of immunization, we will target showing that no efficient A can succeed in $\mathcal{G}_{\mathsf{dist}}$.

Example: the Dual EC BPRG. As an example of a BPRG we turn to Dual EC. It uses an elliptic curve group $\mathbb{G}$ with generator g. For consistency with later sections, we use multiplicative notation for group operations. We also skip for simplicity some details about representation of elliptic curve points, these being unimportant for understanding the attack. For a more detailed description of the algorithm and backdoor see [13].

Key generation K picks a random point $Q \in \mathbb{G}$ and an exponent $d \leftarrow\!\!\$\ \mathbb{Z}_{|\mathbb{G}|}$. It computes $P = Q^d$. The public key is set to $pk = (P, Q)$ and the secret is x. The state space is $\mathcal{S} = \mathbb{Z}_{|\mathbb{G}|}$. On input a seed $s_i \in \mathcal{S}$, the generation algorithm G computes $s_{i+1} \leftarrow P^{s_i}$ and computes r_{i+1} as all but the last 16 bits of $Q^{s_{i+1}}$. The output is (r_{i+1}, s_{i+1}).

With knowledge of d and given two consecutive outputs r_1, r_2 corresponding to states s, s_1 we can give a $\mathcal{G}_{\mathsf{next}}$ adversary A that efficiently recovers s_2. Adversary A starts by computing from r_1 a set of at most 2^{16} possibilities for Q^{s_1}. Let these possibilities be $X_1, \ldots, X_{2^{16}}$. Then for each $i \in [1..2^{16}]$, the adversary checks whether $Q^{X_i^d}$ has all but last 16 bits that match r_2. If so it outputs $s_2 = X_i^d = Q^{s_1 d} = P^{s_1}$. Note that while A cannot recover the generator's second state s_1, it can predict the generator's second output r_2, the third state s_2, and all subsequent states and outputs. Also A is relatively efficient, working in time about 2^{16} operations.

As for basic PRG security without the trapdoor, a result due to Schoenmakers and Sidorenko [25] gives an attack working in time about 2^{16} using a single output to achieve distinguishing advantage around 1/100. Thus, putting it all together, we have that Dual EC is a $(t, q, \delta, (\mathcal{G}_{\mathsf{next}}, 1))$-BPRG for $t \approx 2^{16}$, $q > 2$, and $\delta \approx 1/100$.

From a saboteur's perspective, that Dual EC doesn't achieve PRG security (against distinguishers without the trapdoor) seems a limitation. One can truncate more than 16 bits to achieve better security, but this would make A exponentially less efficient. In the next section we will show how a saboteur can construct a BPRG with strong PRG security and efficient state recovery.

4 Backdoored PRG Constructions

We start by simplifying and improving the Dual EC BPRG. Let G be a group and g a generator of G. Let K pick a random secret key $x \leftarrow\!\!\$\ \mathbb{Z}_{|G|}$ and let $pk \triangleq X = g^x$. The PRG works simply as $\mathrm{G}(pk, s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, X^{s_i})$. A $\mathcal{G}_{\mathsf{next}}$ adversary can recover $s_{i+1} = X^{s_i}$ by computing ${r_{i+1}}^x$. For a G that is DDH secure and for which uniform group elements are indistinguishable from bit strings (e.g., [2,9,29]), this construction can be proven $\mathcal{G}_{\mathsf{dist}}^{\mathrm{PRG}}$ secure under the DDH assumption.

$\underline{\mathrm{K}'}$	$\underline{\mathrm{G}'(pk, s)}$	$\underline{\mathrm{A}'(sk, r_1, \ldots, r_q)}$
$(pk, sk) \leftarrow\!\!\$\ \mathrm{Gen}$	$(r', s') \leftarrow \mathrm{Encap}(pk; s)$	$s' \leftarrow \mathrm{Decap}(sk, r_q)$
return (pk, sk)	**return** (r', s')	**return** s'

Fig. 5. Backdoored PRG from a pseudorandom KEM

4.1 Backdoored PRGs from Key Encapsulation

We in fact can generalize significantly by observing that the previous construction is actually using the ElGamal key encapsulation scheme (KEM) in a direct way. Recall that a KEM scheme triple of algorithms is a KEM $\Gamma = (\mathrm{Gen}, \mathrm{Encap}, \mathrm{Decap})$. The key generation outputs a public key / secret key pair $(pk, sk) \leftarrow \mathrm{Gen}$. The encapsulation algorithm takes the public key, random coins $r \in \{0,1\}^n$ for some n and outputs a ciphertext-key pair $(c, K) \leftarrow \mathrm{Encap}(pk; r)$ where $K \in \{0,1\}^n$. The decapsulation algorithm takes a secret key and ciphertext and outputs a key: $\mathrm{Decap}(sk, c) = \tilde{K} \in \{0,1\}^n \cup \{\mathsf{invalid}\}$. We require correctness, meaning that $\mathrm{Decap}(sk, c) = K$ for $(c, K) = \mathrm{Encap}(pk; r)$ and for all pk, sk pairs generatable by Gen and all coin strings r.

We give a variant of KEM security that requires ciphertexts to be pseudorandom: the output of Encap is indistinguishable from a pair of random bit strings. See Figure 4. We define $\mathsf{Adv}^{\mathrm{Dist}}_{\mathrm{KEM}}(\mathcal{D}) = 2\left|\Pr[\mathrm{Dist}^{\mathcal{D}}_{\mathrm{KEM}} \Rightarrow \mathsf{true}] - \frac{1}{2}\right|$. A KEM Γ is said to be a (t, δ)-pseudorandom KEM if $\mathsf{Adv}^{\mathrm{Dist}}_{\Gamma} \coloneqq \max_{\mathrm{D}} \mathsf{Adv}^{\mathrm{Dist}}_{\Gamma}(\mathrm{D}) \leq \delta$, where the maximum is taken over all algorithms D running in time t.

$\underline{\mathrm{Game}\ \mathrm{Dist}^{\mathcal{D}}_{\mathrm{KEM}}}$
$(pk, sk) \leftarrow\!\!\$\ \mathrm{Gen}$
$r \leftarrow\!\!\$\ \{0,1\}^n$
$(c_0, K_0) \leftarrow\!\!\$\ \mathrm{Encap}(pk; r)$
$c_1 \leftarrow\!\!\$\ \{0,1\}^n$
$K_1 \leftarrow\!\!\$\ \{0,1\}^n$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathrm{D}(pk, c_b, K_b)$
return $(b = b')$

Fig. 4. Pseudorandom KEM security

This is a strictly stronger security notion than the conventional one for KEMs [26], which does not demand that ciphertexts have any particular appearance to attackers. This stronger pseudorandomness requirement was first introduced by Möller [21]. He gave an elliptic curve variant of ElGamal that provably meets it, and other KEM constructions can be built using [2,9,29].

We have the following result showing that any pseudorandom KEM gives a $\mathcal{G}_{\mathsf{next}}$-BPRG.

Proposition 1. *Let* $\Gamma = (\mathrm{Gen}, \mathrm{Encap}, \mathrm{Decap})$ *be a* (t, δ)*-pseudorandom KEM. Then* $(\mathrm{K}', \mathrm{G}', \mathrm{A}')$ *defined in Fig. 5 is a* $(t, q, q\delta, (\mathcal{G}_{\mathsf{next}}, 1))$*-BPRG.*

Proof. The correctness of the KEM gives that $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}', \mathrm{G}', \mathrm{A}') = 1$ and that A' is efficient. We now prove (standard) PRG security against distinguishers without the trapdoor. To do so we use a hybrid argument, applying the pseudorandom KEM security q times. Let H_0 be the $\mathcal{G}^{\mathrm{PRG}}_{\mathsf{dist}}$ game with $b = 0$ and H_q be $\mathcal{G}^{\mathrm{PRG}}_{\mathsf{dist}}$ with $b = 1$. Let H_i for $1 \leq i \leq q-1$ be the same as game H_{i-1} except that we replace the i^{th} output of Encap with two independent, random bit strings. A straightforward reduction gives that $\mathbf{CD}_t(H_i, H_{i+1}) \leq \delta$, and since we have q hybrids $\mathsf{Adv}^{\mathrm{PRG}}_{\mathsf{dist}}(\mathrm{K}, \mathrm{G}, \mathrm{D}) \leq q\delta$ for any D running in time t. □

4.2 Random Seek Backdoored PRGs

We now show a prediction attack, where the prediction algorithm can *seek* to any output quickly. Given one output, we can predict any other, and the prediction can seek in both directions, that is predict previous outputs as well. In the construction shown, we use the lsb of a random string to make a decision, and we shift by one bit, so that the randomness used later is independent of the bit used for decision. We assume that the underlying PRG or PRF was used to get enough number of bits so that after the shift we have enough random bits for encryption.

Proposition 2. *Let* $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$ *be a* (t, q, δ)-$\mathsf{IND\$\text{-}CPA}$ *public key encryption scheme,* F_{sk} *be a* (t, q, δ)*-pseudorandom function. Then* $(\mathrm{K}', \mathrm{G}', \mathrm{A}')$ *defined in Fig. 6 is a* $(t, q, 3\delta, (\mathcal{G}_{\mathsf{rseek}}, \frac{1}{4} - \delta))$*-backdoored pseudorandom generator.*

$\underline{\mathrm{K}'}$	$\underline{\mathrm{G}'(pk, (s_0, s_1, \mathsf{count}))}$	$\underline{\mathrm{A}'(sk, i, j, r_i)}$
$(pk, sk) \leftarrow \mathrm{K}$	$\alpha \leftarrow \mathrm{F}_{s_1}(\mathsf{count})$	**if** $(i = j)$ **then**
return (pk, sk)	**if** $\mathrm{lsb}(\alpha) = 0$ **then**	**return** r_i
	$r \leftarrow \mathrm{Enc}_{pk}(s_0; \alpha^{\gg 1})$	$s_0 \leftarrow \mathrm{Dec}_{sk}(r_i)$
	else	**if** $(s_0 = \perp)$ **then**
	$r \leftarrow \mathrm{F}_{s_0}(\mathsf{count})$	**return** 0
	$\mathsf{count} \leftarrow \mathsf{count} + 1$	$r_j \leftarrow \mathrm{F}_{s_0}(j)$
	return $(r, (s_0, s_1, \mathsf{count}))$	**return** r_j

Fig. 6. Random seek backdoored PRG

Proof.

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{rseek}}(\mathrm{K}', \mathrm{G}', \mathrm{A}') = \Pr[r_j = r'_j] \geq \Pr[\mathrm{lsb}(\mathrm{F}_{s1}(i)) = 0 \wedge \mathrm{lsb}(\mathrm{F}_{s1}(j)) = 1] \geq \frac{1}{4} - \delta.$$

From pseudorandomness of F's outputs

$$\mathbf{CD}_t(\mathrm{F}_{s_0}(1), \ldots, \mathrm{F}_{s_0}(q), \mathcal{U}) \leq \delta, \mathbf{CD}_t(\mathrm{F}_{s_1}(1), \ldots, \mathrm{F}_{s_1}(q), \mathcal{U}) \leq \delta.$$

Then $\mathbf{CD}_t((pk, s_0, \mathrm{Enc}_{pk}(s_0; \alpha^{\gg 1})), (pk, s_0, \mathcal{U})) \leq 2\delta$ due to IND$-CPA security. Thus,

$$\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}'_{pk}, \mathcal{U})), \mathcal{U}) \leq 3\delta.$$

□

The distinguishing and predicting PRGs we discussed also satisfy the notion of forward security, whereas the $\mathcal{G}_{\mathsf{rseek}}$ construction in Fig 6 is forward-insecure.

4.3 Public-Key Encryption from a Backdoor PRG

We show that the existence of backdoored PRGs implies public-key encryption (PKE). From a backdoored PRG, we construct a bit encryption scheme with noticeable correctness and overwhelming secrecy. Using parallel repetition and privacy amplification of key-agreement [18], we can amplify secrecy and correctness without increasing the number of rounds. Since the number of rounds is not increased, we obtain secure public-key encryption.

Theorem 1. *If* $(\mathrm{K}, \mathrm{G}_{pk}, \mathrm{A})$ *is a* $(t, q, \delta, (\mathcal{G}_{\mathsf{dist}}, \varepsilon))$*-backdoored PRG, then the protocol in Fig. 7 is a bit-encryption protocol with correctness* ε *and security* $1 - \delta$ *against attackers running in time* t*.*

$\underline{\mathrm{Gen}}$	$\underline{\mathrm{Enc}(pk, b)}$	$\underline{\mathrm{Dec}(sk, r)}$
$(pk, sk) \leftarrow \mathrm{K}$	$s \leftarrow \mathcal{U}$	$b' \leftarrow \mathrm{A}(sk, r)$
return (pk, sk)	**if** $(b = 0)$ **then**	**return** b'
	$\quad r \leftarrow \mathcal{U}$	
	else	
	$\quad r \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$	
	return r	

Fig. 7. Bit Encryption

Proof. orrectness:

$$\Pr[\mathrm{Dec}(sk, \mathrm{Enc}(pk, b)) = b] = \Pr[b = b'] = \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, \mathrm{G}, \mathrm{A}) \Rightarrow \mathsf{true}] \geq \frac{1}{2} + \frac{\varepsilon}{2}.$$

Security:
For any adversary D who runs in time t,

$$\Pr[\mathrm{D}(pk, r) = b] = \frac{1 + \mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}, \mathcal{U})), \mathcal{U})}{2} \leq \frac{1}{2} + \frac{\delta}{2}.$$

□

Note that combining this result with our earlier construction of backdoored PRGs from PKE and Proposition 1, we arrive at the promised conclusion that backdoored PRGs and pseudorandom PKE are equivalent. We capture this with the following informal theorem, which is a corollary of the results so far.

Theorem 2. *Backdoor PRGs exist iff public-key encryption with pseudorandom ciphertexts exists.*

5 Immunization

In this section, we ask how to *immunize* a potentially backdoored PRG. A natural idea is for the user to apply a non-trivial function f to the output of the PRG. So now the attacker A learns $f(r_i)$ rather than r_i. We ask the question: when does f successfully immunize a PRG? We study the immunization functions that turn a backdoored PRG into a backdoor-less PRG. Letting the immunization function be a family of algorithms $\{f_{\mathsf{seed}} \,|\, \mathsf{seed} \in \{0,1\}^\ell\}$, we consider the following immunization models:

1. Public immunization: In this model, seed is revealed to the attacker A prior to construction of the PRG algorithm. The attacker thus knows the immunization function f_{seed} that will be applied to the outputs of the generator. In this setting, the goal of the attacker A is to develop a PRG G with a backdoor that bypasses the known immunization.
2. Semi-private immunization: In this model, the PRG generator G is constructed without reference to seed. We may view this as a setting in which the PRG attacker A learns seed, and thus f_{seed}, only *after* the specification of G. This situation can arise, for example, when the immunization function f depends upon a source of fresh public randomness.
3. Private immunization: In this model, seed is secret, in the sense that G is constructed without reference to seed and A never learns seed. We might imagine the user using a source of private randomness, unavailable to A, to seed the immunization function f. (Note that although the user has some private randomness, she might still need the PRG to generate longer pseudorandom strings.)

Now we give formal definitions of secure immunization in the three models discussed above. We slightly abuse notation in the following way: For a PRG G such that $(r_i, s_i) \leftarrow \mathrm{G}(s_{i-1})$, we write $f \circ \mathrm{G}$ to mean $f(r_i)$, i.e., f applied to the output of G only (and not G's internal state). Similarly, by $\mathsf{out}^q(f \circ \mathrm{G}, s)$ we mean the sequence $(f(r_1), \ldots, f(r_q))$, where $(r_1, \ldots, r_q) = \mathsf{out}^q(\mathrm{G}, s)$.

Definition 7 (Public Immunization). *Let* $\mathsf{type} \in \{\mathsf{dist}, \mathsf{next}, \mathsf{rseek}\}$. *A family of algorithms* $\{f_{\mathsf{seed}} \,|\, \mathsf{seed} \in \{0,1\}^\ell\}$ *is called a* public $((t,q,\delta),(t',q',\delta'),(\mathcal{G}_{\mathsf{type}},\varepsilon))$-immunization*, if for any* (t,q,δ)*-secure PRG* (K,G)*, and for any algorithm* A *running in time* t'*,*

- $\{f_{\mathsf{seed}} \circ \mathrm{G}_{pk}(\mathsf{seed},\cdot) \,|\, (\mathsf{seed}, pk) \in \{0,1\}^\ell \times \{0,1\}^p\}$ *is a* (t',q',δ')*-pseudorandom generator,*

– $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{type}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}(\mathsf{seed}, \cdot), \mathrm{A}(\mathsf{seed}, \cdot)) \leq \varepsilon$.

Definition 8 (Semi-private Immunization). *Let* $\mathsf{type} \in \{\mathsf{dist}, \mathsf{next}, \mathsf{rseek}\}$. *A family of algorithms* $\{f_{\mathsf{seed}} \,|\, \mathsf{seed} \in \{0,1\}^{\ell}\}$ *is called a* semi-private $((t,q,\delta),(t',q',\delta'),(\mathcal{G}_{\mathsf{type}},\varepsilon))$-immunization*, if for any* (t,q,δ)*-secure PRG* (K,G)*, and for any algorithm* A *running in time* t'*,*

– $\{f_{\mathsf{seed}} \circ \mathrm{G}_{pk}(\cdot) \,|\, (\mathsf{seed}, pk) \in \{0,1\}^{\ell} \times \{0,1\}^{p}\}$ *is a* (t',q',δ')*-pseudorandom generator,*
– $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{type}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}(\cdot), \mathrm{A}(\mathsf{seed}, \cdot)) \leq \varepsilon$.

Definition 9 (Private Immunization). *Let* $\mathsf{type} \in \{\mathsf{dist}, \mathsf{next}, \mathsf{rseek}\}$. *A family of algorithms* $\{f_{\mathsf{seed}} \,|\, \mathsf{seed} \in \{0,1\}^{\ell}\}$ *is called a* private $((t,q,\delta),(t',q',\delta'),(\mathcal{G}_{\mathsf{type}},\varepsilon))$-immunization*, if for any* (t,q,δ)*-secure PRG* (K,G)*, and for any algorithm* A *running in time* t'*,*

– $\{f_{\mathsf{seed}} \circ \mathrm{G}_{pk}(\cdot) \,|\, (\mathsf{seed}, pk) \in \{0,1\}^{\ell} \times \{0,1\}^{p}\}$ *is a* (t',q',δ')*-pseudorandom generator,*
– $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{type}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}(\cdot), \mathrm{A}(\cdot)) \leq \varepsilon$.

In Section 5.1 we show that it is possible to successfully create a PRG backdoor in the public immunization setting. On the other hand, in Section 5.2 we show that there exist immunizations in the semi-private model that separate these two models. Also, as we will see in Section 5.3, a pseudorandom function is a secure private immunization. To separate semi-private and private models, in the following simple lemma we show that a pseudorandom permutation is not a semi-private immunization. The construction we give for the separation in Fig. 8 also satisfies forward security.

Lemma 1. *Let* f_{seed} *be a* (t,q,δ)*-pseudorandom permutation. Then there exists a triple* $(\mathrm{K}',\mathrm{G}',\mathrm{A}')$*, such that* $(\mathrm{K}', f_{\mathsf{seed}} \circ \mathrm{G}'(\cdot), \mathrm{A}'(\mathsf{seed}, \cdot))$ *is a* $(t,q,2q\delta,(\mathcal{G}_{\mathsf{next}},1))$*-backdoored pseudorandom generator.*

Proof. Let G be a (t',q,δ)-pseudorandom generator, and $\Gamma = (\mathrm{Gen},\mathrm{Encap},\mathrm{Decap})$ be a (t,δ)-pseudorandom ciphertext KEM. Consider the triple $(\mathrm{K}', \mathrm{G}'_{pk}, \mathrm{A}'(\mathsf{seed},\cdot))$ shown in Fig. 8. Then $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}', f_{\mathsf{seed}} \circ \mathrm{G}', \mathrm{A}') = \Pr[\mathrm{Decap}(r', sk) = s' | (r', s') \leftarrow \mathrm{Encap}(pk;\alpha), (pk, sk) \leftarrow \mathrm{Gen}] = 1$. From pseudorandomness of G's outputs $\mathbf{CD}_t(\mathsf{out}^q(\mathrm{G},\mathcal{U}),\mathcal{U}) \leq \delta$, (t,δ)-pseudorandomness of Γ, and using a hybrid argument similar to the proof of Proposition 1,

$$\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}'_{pk},\mathcal{U})),\mathcal{U}) \leq 2q\delta.$$

□

5.1 Public Immunization Model

In the public immunization model, the PRG algorithms G and A know the seed of the immunization function that will be applied on the output. In this section

$\underline{\mathrm{K}'}$
$(pk, sk) \leftarrow \mathrm{Gen}$
return (pk, sk)

$\underline{\mathrm{G}'(pk, s)}$
$(\alpha, \beta) \leftarrow \mathrm{G}(s)$
$(r', s') \leftarrow \mathrm{Encap}(pk; \alpha)$
return (r', s')

$\underline{\mathrm{A}'(sk, \mathsf{seed}, f_{\mathsf{seed}}(r_i))}$
$c \leftarrow f_{\mathsf{seed}}^{-1}(f_{\mathsf{seed}}(r_i))$
$s' \leftarrow \mathrm{Decap}(c, sk)$
return s'_i

Fig. 8. PRP Immunization Insecure in Semi-private Model

we demonstrate backdoored pseudorandom generator that cannot be immunized in the public immunization model. Since seed of the immunization function f_{seed} is known to both G and A, in order to construct a backdoored pseudorandom generator from the viewpoint of the saboteur, we fix the strongest function from this family so that for any function in the family, the backdoored PRG after immunization is compromised. The idea behind the construction is to leak the initial state bit by bit, by rejection sampling of the output of the immunization function such that the bias is unnoticeable. For a bit string s, we denote the ith bit of s by $s_{(i)}$.

$\underline{\mathrm{K}'}$
$(pk, sk) \leftarrow \mathrm{K}$
return (pk, sk)

$\underline{\mathrm{G}'(pk, \mathsf{seed}, (s_0, s_1, \mathsf{count}))}$
$c \leftarrow \mathrm{Enc}_{pk}(s_1)$
$L \leftarrow |c|$
if $\mathsf{count} \leq L$ **then**
  $j \leftarrow 0$
  $s'_0 \leftarrow s_0$
  $\mathsf{count}_2 \leftarrow 0$
  repeat
    $\mathsf{count}_2 \leftarrow \mathsf{count}_2 + 1$
    $(r', s'_0) \leftarrow \mathrm{G}(s'_0)$
  until $(f_{\mathsf{seed}}(r')_{(1)} = c_{(\mathsf{count})}) \vee (\mathsf{count}_2 > \frac{\ln L}{1-\delta})$;
  $s'_1 \leftarrow s_1$
else
  $(r', s'_1) \leftarrow \mathrm{G}(s_1)$
  $s'_0 \leftarrow 0$
$\mathsf{count}' \leftarrow \mathsf{count} + 1$
return $(r', (s'_0, s'_1, \mathsf{count}'))$

$\underline{\mathrm{A}'(sk, \mathsf{seed}, f_{\mathsf{seed}}(r_1), \ldots, f_{\mathsf{seed}}(r_q))}$
for $1 \leq i \leq L$ **do**
  $c_i \leftarrow f_{\mathsf{seed}}(r_i)_{(1)}$
$s_1 \leftarrow \mathrm{Dec}_{sk}(c)$
for $L + 1 \leq i \leq q$ **do**
  $r'_i, s_1 \leftarrow \mathrm{G}(s_1)$
return $(0, s_1, q + 1)$

Fig. 9. Predicting backdoored PRG in Public Model

Lemma 2. *Let* $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$ *be a* $(t, q, \delta) - \mathsf{IND\$\text{-}CPA}$ *public key encryption scheme,* G *be a* (t, q, δ)*-pseudorandom generator, and* f *such that for* $\mathsf{seed} \leftarrow \{0,1\}^{\ell}$, $\mathbf{CD}_t(f_{\mathsf{seed}}(\mathcal{U}), \mathcal{U}) \leq \delta$. *Then* $(\mathrm{K}', \mathrm{G}', \mathrm{A}')$ *defined in Fig. 9 is a* $(t, q - L\frac{\ln L}{1-\delta}, 2\delta, (\mathcal{G}_{\mathsf{next}}, \varepsilon))$*-backdoored pseudorandom generator, where* $\varepsilon = 1 - L \cdot \exp\left(\frac{-\ln^2 L}{3(1-\delta)}\right)$, L *is the length of ciphertexts produced by* Enc_{pk}.

Proof. From pseudorandomness of G's outputs $\mathbf{CD}_t(\mathsf{out}^q(\mathrm{G},\mathcal{U}),\mathcal{U}) \leq \delta$ and pseudorandomness of ciphertexts $\mathbf{CD}_t((pk, s_1, \mathrm{Enc}_{pk}(s_1)), (pk, s_1, \mathcal{U})) \leq \delta$,

$$\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}'_{pk}, \mathcal{U})), \mathcal{U}) \leq 2\delta.$$

From the Chernoff bound:

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}', \mathrm{G}', \mathrm{A}') \geq 1 - L \cdot \Pr[\mathsf{count}_2 \geq \frac{\ln L}{1-\delta}] \geq 1 - L \cdot \exp\left(\frac{-\ln^2 L}{3(1-\delta)}\right).$$

□

5.2 Semi-Private Immunization Model

In the semi-private model, the generator G does not know seed of f_{seed}, but the attacker does. We show that a Random Oracle and a Universal Computational Extractor are secure immunizations in the semi-private model. We will first bound the collision probability of pseudorandom outputs. The collision probability bounds the probability that an algorithm can predict the output of a PRG run on a uniformly random seed, even with the knowledge of some trapdoor information, because, intuitively, the output of a PRG should depend on the input seed also.

Definition 10. *The* conditional predictability *of X conditioned on Y is defined as*

$$\mathrm{Pred}(X|Y) \doteq \mathrm{E}_{y \leftarrow Y}[\max_x(\Pr[X = x | y = Y])].$$

Definition 11. *The* conditional collision probability *of X conditioned on Y is defined as*

$$\mathrm{Col}(X|Y) \doteq \mathrm{E}_{y \leftarrow Y}[\Pr_{x_1, x_2 \leftarrow X}[x_1 = x_2 | Y = y]].$$

Lemma 3. *For any distributions X and Y,* $\mathrm{Pred}(X|Y) \leq \sqrt{\mathrm{Col}(X|Y)}$.

Proof. Let $p_y = \Pr[Y = y], p_{x|y} = \Pr[X = x | Y = y]$. Then

$$\mathrm{Pred}[X|Y] = \sum_y p_y \cdot \max_x p_{x|y} = \sum_y \sqrt{p_y} \cdot (\sqrt{p_y} \max_x p_{x|y}) \leq$$

$$\sqrt{\sum_y p_y \cdot \sum_y p_y \max p^2_{x|y}} \leq \sqrt{1 \cdot \sum_y (p_y \cdot \sum_x p^2_{x|y})} = \sqrt{\mathrm{Col}(X|Y)}.$$

□

Let $\{\mathrm{G}_{pk} \colon \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^m | pk \in \{0,1\}^p\}$ be a family of algorithms, then by $\mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U})$ we denote the distribution of G_{pk}'s ith output, i.e. the distribution of r_i where $(r_1, \ldots, r_i, \ldots, r_q) \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U})$.

Lemma 4. *Let* $\{\mathrm{G}_{pk}\colon \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^m | pk \in \{0,1\}^p\}$ *be a* (t,q,δ)*-pseudorandom generator.* [1] *Then for any* $1 \le i \le q$*, for any* $\mathrm{K} \to \{0,1\}^p \times \{0,1\}^k$ *such that* $\mathbf{CD}_t(pk, U_p) \le \delta$*, where* $(pk, sk) \leftarrow \mathrm{K}$*,*

$$\mathrm{Pred}(\mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U}) | sk) \le \sqrt{\delta + \frac{1}{2^n}}.$$

Proof. We show that $\mathrm{Col}(\mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U}) | sk) \le \delta + \frac{1}{2^n}$, then Lemma 3 implies the desired bound.

Assume, to the contrary, $\mathrm{Col}(\mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U}) | sk) > \delta + \frac{1}{2^n}$. This implies that there exists i such that $\mathrm{E}_{(pk,sk)\leftarrow \mathrm{K}} \Pr[r_i = r_i' | sk] > \delta + \frac{1}{2^n}$, where $r_i, r_i' \leftarrow \mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U})$. Let D be a PRG-distinguisher for G_{pk} as defined in Fig. 10. Then,

$$|\Pr[\mathrm{D}(\mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U})) = 1] - \Pr[\mathrm{D}(U) = 1]| \ge \mathrm{Col}(\mathsf{out}_i(\mathrm{G}_{pk}, \mathcal{U}) | sk) - \frac{1}{2^n} > \delta,$$

which contradicts the (t,q,δ)-pseudorandomness of $\{\mathrm{G}_{pk}\}$. □

$\underline{\mathrm{D}(pk, r_1, \ldots, r_q)}$
$s \leftarrow \{0,1\}^m$
$r_1', \cdots, r_q' \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$
if $r_i = r_i'$ **then**
  return 1
else
  return 0

Fig. 10. Distinguisher D for G_{pk}

Positive Result in Random Oracle Model. A random oracle (RO) is an oracle that responds to every unique query with a random response chosen uniformly from its output domain. If a query is repeated it responds the same way every time that query is submitted. A RO : $\{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ is chosen uniformly at random from the set of all functions that map $\{0,1\}^{n+k}$ to $\{0,1\}^n$. We show that in the semi-private model, a Random Oracle is a secure immunization function.

Theorem 3. *Let* $\{\mathrm{G}_{pk}\colon \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^m | pk \in \{0,1\}^p\}$ *be a* (t,q,δ)*-pseudorandom generator. Then* $f_{\mathsf{seed}}(x) = \mathrm{RO}(x \| \mathsf{seed})$ *is a semiprivate* $((t,q,\delta),(t,q,\delta),(\mathcal{G}_{\mathsf{dist}},\varepsilon))$*-immunization for* G_{pk}*, where*

$$\varepsilon = \delta + \frac{q^2}{2^n} + \frac{q_G}{2^k} + q q_{\mathrm{A}} \sqrt{\delta + \frac{1}{2^n}},$$

[1] Here and below we assume that $t > C(p + q(n + m + \mathsf{time}(\mathrm{G}_{pk})))$ for some absolute constant $C > 0$, so that the attacker can always parse the input, and run G for q times.

$k = |\mathsf{seed}|$, q_{G} and q_{A} are the bounds on the number of times G *and* A *query the random oracle, respectively.*

Proof. Assume, to the contrary, there exists a pair of algorithms (K, A) running in time t', such that the triple $(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}(\cdot), \mathrm{A}(\mathsf{seed}, \cdot))$ is a $(t, q, \delta, (\mathcal{G}_{\mathsf{dist}}, \varepsilon))$-backdoored pseudorandom generator. I.e.,

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}, \mathrm{A}) = 2\left|\Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}, \mathrm{A}) \Rightarrow \mathsf{true}] - \frac{1}{2}\right| > \varepsilon$$

in Game $\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}, \mathrm{A})$ from Fig. 3. Let $r_1, \ldots, r_q$ be the outputs of G_{pk} before the immunization, i.e. $s \leftarrow \mathcal{U}, (r_1, \ldots, r_q) \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$. The immunization is $f_{\mathsf{seed}}(r_i) = \mathrm{RO}(r_i \| \mathsf{seed})$ for $1 \le i \le q$.

We define the following three events:

- W_1: $r_i = r_j$ for $i \neq j$.
- W_2: G_{pk} queries $(r_i \| \mathsf{seed})$ for some $1 \le i \le q$.
- W_3: A queries $(r_i \| \mathsf{seed})$ for some $1 \le i \le q$.

Note that if none of the events above happened then the two distributions in the distinguishing game corresponding to the challenge bit being 0 or 1, are identical. Now we proceed to bound the probabilities of these three events.

- Since the PRG-security of G is δ, $\Pr[W_1] \le \frac{q^2}{2^n} + \delta$.
- In the semiprivate model G does not see seed, therefore, the probability that G queries $r_i \| \mathsf{seed}$ in one of its queries is the probability that the G guesses seed, and by the union bound this is bounded from above by $\frac{q^{\mathrm{G}}}{2^k}$. Thus, $\Pr[W_2] \le q^{\mathrm{G}}/2^k$.
- Now, we look at the probability that A makes a relevant query, given that G did not query $r_i \| \mathsf{seed}$ for all i. Assume A predicts r_i for $i \in I \subseteq [q]$. Then there exists $i \in I$ that was predicted first, i.e. when all $f_{\mathsf{seed}}(r_j)$ looked random to A. Then, the probability that A predicts at least one r_i is at most $\sum_{i=1}^{q} \Pr[\mathrm{A} \text{ predicts } r_i \text{ using } q_{\mathrm{A}} \text{ queries given } sk]$. Since A makes at most q_{A} calls to the random oracle, the latter probability, by the union bound, is bounded by $q_{\mathrm{A}} \sum_{i=1}^{q} \Pr[\mathrm{A} \text{ predicts } r_i \text{ using one query given } sk]$. Now Lemma 4 gives us the following bound:

$$\begin{aligned}
\Pr[W_3] &\le \sum_{i=1}^{q} \Pr[\mathrm{A} \text{ predicts } r_i \text{ using } q_{\mathrm{A}} \text{ queries} | sk] \\
&\le q_{\mathrm{A}} \sum_{i=1}^{q} \Pr[\mathrm{A} \text{ predicts } r_i \text{ using one query} | sk] \\
&\le q_{\mathrm{A}} \sum_{i=1}^{q} \mathrm{Pred}[r_i | sk] \le q q_{\mathrm{A}} \sqrt{\delta + \frac{1}{2^n}}.
\end{aligned}$$

By the claims above,

$$\varepsilon = \Pr[W_1] + \Pr[W_2] + \Pr[W_3] \leq \delta + \frac{q^2}{2^n} + \frac{q_{\mathrm{G}}}{2^k} + qq_{\mathrm{A}}\sqrt{\delta + \frac{1}{2^n}}.$$

□

Positive Result in Standard Model. In this section, we show that replacing the Random Oracle with a UCE function [4] is secure in the standard model. First, we briefly recall Universal Computational Extractor (UCE) defined in [4] by Bellare et al. UCE is a family of security notions for a hash function family.

UCE Security. A notion of UCE security is specified by specifying a class of sources $\mathcal{S}$. The source is given oracle access to the hash function. UCE security for the class of sources $\mathcal{S}$ states that for any PPT algorithm called the distinguisher D, who receives the key of the hash function and leakage L passed by the source, cannot tell better than random guessing whether H_k was used or a random function. We now give the formal definitions. A source S is a PPT algorithm which is given oracle access to Hash, and outputs a value L called the leakage. For a pair of source S and distinguisher D, define the $\mathrm{UCE}_H^{S,D}$ game as shown in Fig. 11.

Definition 12. *A function H is called* $\mathrm{UCE}[\mathcal{S}, q_{\mathrm{D}}, \epsilon]$-secure*, if for all sources $S \in \mathcal{S}$, and all polynomial-time algorithms* D *that make at most q_{D} queries to H,* $\mathsf{Adv}_H^{\mathrm{UCE}}(S, D) := 2\Pr[\mathrm{UCE}_H^{S,D} \Rightarrow \mathsf{true}] - 1 \leq \epsilon$.

For a source S, and a polynomial-time algorithm P called the predictor, define the game Pred_S^P as shown in Fig. 12.

Definition 13. *A source S is called* (l, ϵ)-statistically unpredictable*, denoted by $S \in \mathcal{S}^{sup}[l, \epsilon]$, if for all computationally unbounded algorithms P that output a list of at most l guesses* $\mathsf{Adv}_{S,P}^{\mathrm{Pred}} := \Pr[\mathrm{Pred}_S^P \Rightarrow \mathsf{true}] \leq \epsilon$.

Main $\mathrm{UCE}_H^{S,D}$
$b \leftarrow \{0,1\}$
$k \leftarrow \mathrm{K}$
$L \leftarrow S^{Hash}$
$b' \leftarrow D(k, L)$
return $(b' = b)$

Hash(x)
if $T[x] = \perp$ **then**
  if $(b = 1)$ **then**
    $T[x] \leftarrow H(k, x)$
  else
    $T[x] \leftarrow \{0,1\}^k$
return $T[x]$

Fig. 11. Game UCE and Hash Oracle

$\underline{\text{Pred}_S^P}$
done $\leftarrow$ false
$Q \leftarrow \emptyset$
$L \leftarrow S^{Hash}$
done $\leftarrow$ true
$Q' \leftarrow P^{Hash}(L)$
return $(Q \cap Q' \neq \emptyset)$

$\underline{\text{Hash}(x)}$
if done $\leftarrow$ false **then**
  $Q \leftarrow Q \cup \{x\}$
if $T[x] = \perp$ **then**
  $T[x] \leftarrow \{0,1\}^k$
return $T[x]$

Fig. 12. Game Pred and Hash Oracle

Theorem 4. *Let* $\{\mathrm{G}_{pk}\colon \{0,1\}^m \rightarrow \{0,1\}^n \times \{0,1\}^m | pk \in \{0,1\}^p\}$ *be a* (t,q,δ)*-pseudorandom generator. Then* $f_{\mathsf{seed}}(x) = H_{\mathsf{seed}}(x)$ *is a semiprivate* $((t,q,\delta),(t,q,\delta+\varepsilon),(\mathcal{G}_{\mathsf{dist}},\varepsilon'))$*-immunization for* G_{pk}*, where*

$$\varepsilon' = 2\epsilon + \delta + \frac{q^2}{2^n},$$

$H \in \mathrm{UCE}[\mathcal{S}, q_D, \epsilon]$, $\mathcal{S} = \mathcal{S}^{sup}[l, \delta + \frac{q^2}{2^n} + ql\sqrt{\delta + \frac{1}{2^n}}]$.

Proof. Given an adversary A playing the distinguishing attack game $\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, H \circ \mathrm{G}, \mathrm{A}(\mathsf{seed}))$ we will construct a statistically unpredictable source S and a polynomial-time distinguisher D (see Fig. 13) such that $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, H \circ \mathrm{G}, \mathrm{A}(\mathsf{seed})) \leq 2\mathsf{Adv}^{\mathrm{UCE}}_H(S, D) + \delta + \frac{q^2}{2^n}$.

$\underline{S^{\mathrm{Hash}}}$
$(pk, sk) \leftarrow \mathrm{K}$
$s \leftarrow \{0,1\}^m$
$r_1, r_2, \cdots, r_q \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, s)$
for $1 \leq i \leq q$ **do**
  $u_i^0 \leftarrow \mathrm{Hash}(R_i)$
  $u_i^1 \leftarrow \{0,1\}^n$
$d \leftarrow \{0,1\}$
$I = \{u_1^d, \ldots, u_q^d\}$
return (d, pk, sk, I)

$\underline{\mathrm{D}(d, sk, I, k)}$
$d' \leftarrow \mathrm{A}(sk, I, k)$
if $(d = d')$ **then**
  return 1
else
  return 0

Fig. 13. Source S and Distinguisher D

Let b be the challenge bit in the UCE game $\mathrm{UCE}^{S,\mathrm{D}}_H$. Then,

$$\Pr[\mathrm{UCE}^{S,D}_H \Rightarrow \mathsf{true} | b = 1] = \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, H \circ \mathrm{G}, \mathrm{A}(\mathsf{seed})) \Rightarrow \mathsf{true}],$$
$$\Pr[\mathrm{UCE}^{S,\mathrm{D}}_H \Rightarrow \mathsf{true} | b = 0] = 1 - \Pr[\mathcal{G}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, \mathrm{RO} \circ \mathrm{G}, \mathrm{A}(\mathsf{seed})) \Rightarrow \mathsf{true}],$$

where in the RO immunization game, A has to distinguish uniformly random outputs from RO applied to the outputs of G. If r's are distinct, then these two distributions are identical. From the PRG security, the probability of the event $r_i = r_j$ for $i \neq j$ is less than $\delta + \frac{q^2}{2^n}$. Therefore,

$$\Pr[\mathrm{UCE}_H^{S,\mathrm{D}} \Rightarrow \mathsf{true} | b = 0] \geq \frac{1}{2} - \frac{1}{2}(\delta + \frac{q^2}{2^n})$$

Summing yields,

$$\mathsf{Adv}_H^{\mathrm{UCE}}(S, \mathrm{D}) = \frac{1}{2}\mathsf{Adv}_{\mathsf{dist}}^{\mathrm{BPRG}}(\mathrm{K}, H \circ \mathrm{G}, \mathrm{A}) - \frac{1}{2}\delta - \frac{1}{2}.\frac{q^2}{2^n},$$
$$\mathsf{Adv}_{\mathsf{dist}}^{\mathrm{BPRG}}(\mathrm{K}, H \circ \mathrm{G}, \mathrm{A}) \leq 2\mathsf{Adv}_H^{\mathrm{UCE}}(S, D) + \delta + \frac{q^2}{2^n}.$$

Now we argue that S is statistically unpredictable; that is, it is hard to guess the source's Hash queries even given the leakage, in the random case of the UCE game. Consider an arbitrary predictor P, and the advantage of P in the game Pred_S^P. If all R_i are distinct (which happens with probability $1 - \delta - \frac{q^2}{2^n}$), the probability that P guesses at least one of r's given the leakage is at most $q\mathrm{Pred}(R|sk)$. Now, since P outputs a list of length l, by Lemma 4,

$$\mathsf{Adv}_{S,P}^{\mathrm{Pred}} \leq \delta + \frac{q^2}{2^n} + ql\sqrt{\delta + \frac{1}{2^n}}.$$

□

5.3 Private Immunization Model

We now study the strongest model of immunization which is the private model, where seed is secret from both the PRG and the attacker. We show that a PRF is an immunization function in this model. But if users had access to a backdoor-less PRF, then instead of using it to immunize a backdoored PRG, they could use the PRF itself for pseudorandomness. In this section, we explore using functions weaker than PRF as immunization functions, and show that some natural functions are not secure immunizations.

PRF Immunization

Lemma 5. *Let* $\{\mathrm{G}_{pk} \colon \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^m | pk \in \{0,1\}^p\}$ *be a* (t, q, δ)*-pseudorandom generator, let also* $\{f_{\mathsf{seed}} \colon \{0,1\}^n \to \{0,1\}^k | \mathsf{seed} \in \{0,1\}^l\}$ *be a* (t, q, ε)*-pseudorandom function. Then* f_{seed} *is a private* $((t, q, \delta), (t, q, \delta + \varepsilon), (\mathcal{G}_{\mathsf{dist}}, \varepsilon'))$*-immunization for* G_{pk}*, where*

$$\varepsilon' = \varepsilon + \delta + \frac{q^2}{2^n}.$$

Proof. From the definition of PRF, no distinguisher D running in time t given q outputs of F_{seed} can distinguish the output from uniformly random with advantage greater than ε. By PRG security of G_{pk}, $\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U})), \mathcal{U}) \leq \delta$. Therefore, $\{f_{\mathsf{seed}} \circ \mathrm{G}_{pk}(\cdot) | (\mathsf{seed}, pk) \in \{0,1\}^\ell \times \{0,1\}^p\}$ is a $(t, q, \delta + \varepsilon)$-pseudorandom generator. Similar to the proof of Theorem 3, $\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{dist}}(\mathrm{K}, f_{\mathsf{seed}} \circ \mathrm{G}(\cdot), \mathrm{A}(\cdot)) \leq \mathsf{Adv}^{\mathrm{PRF}}_{f} + \Pr[\exists i, j : r_i = r_j | (r_1, \ldots, r_q) \leftarrow \mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U})] \leq \varepsilon + \delta + \frac{q^2}{2^n}$. □

Attack against XOR. One of natural candidates for the immunization function in the private randomness model is the function XOR with a random string as private randomness. In this section we show an attack against $f_{\mathsf{seed}}(x) = \mathsf{seed} \oplus x$, where seed is private randomness of immunization function f. The backdoored PRG works as follows: it outputs strings such that the XOR of two consecutive outputs leaks one bit of s_1 where s_1 is a part of the seed of length n, such that the bias introduced is negligible. After $(n+1)$ outputs A can recover all of s_1, and can predict future outputs.

Lemma 6. *Let* $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$ *be a* $(t, q, \delta) - \mathsf{IND\$\text{-}CPA}$ *public key encryption scheme,* G *be a* (t, q, δ)*-pseudorandom generator. Then for* $(\mathrm{K}', \mathrm{G}', \mathrm{A}')$ *defined in Fig. 14 and* $f_{\mathsf{seed}}(x) = \mathsf{seed} \oplus x$, $(\mathrm{K}', f_{circ} \circ \mathrm{G}'(\cdot), \mathrm{A}'(\cdot))$ *is a* $(t, q - n\frac{\log n}{1-\delta}, 2\delta, (\mathcal{G}_{\mathsf{next}}, \varepsilon))$*-backdoored pseudorandom generator, where* $\varepsilon = 1 - n \cdot \exp\left(\frac{-\ln^2 n}{3(1-\delta)}\right)$.

Proof. From the Chernoff bound:

$$\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{next}}(\mathrm{K}', \mathrm{G}', \mathrm{A}') \geq 1 - n \cdot \Pr[\mathsf{count}_2 \geq \frac{\ln n}{1-\delta}] \geq 1 - n \cdot \exp\left(\frac{-\ln^2 n}{3(1-\delta)}\right).$$

From pseudorandomness of G's outputs $\mathbf{CD}_t(\mathsf{out}^q(\mathrm{G}_{pk}, \mathcal{U}), \mathcal{U}) \leq \delta$, and $\mathbf{CD}_t((pk, s_1, \mathrm{Enc}(s_1)), (pk, s_1, \mathcal{U})) \leq \delta$ due to $\mathsf{IND\$\text{-}CPA}$ security. Thus, $\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}'_{pk}, \mathcal{U})), \mathcal{U}) \leq 2\delta$. □

Extensions of XOR-attack. The previous attack can be extended in two ways. First, the PRG can be modified so that one does not need to see $q > n$ outputs to guess the next one, with high probability it is enough to see just three random outputs. Although this kind of attack is weaker than the definition of rseek-attack, it is much stronger than next-attack. Second, using homomorphic encryption, the previous attack can be extended to some other natural immunization functions. Here we show an example where the multiplication with a private random string is not a private immunization. Let $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$ be a homomorphic $(t, q, \delta) - \mathsf{IND\$\text{-}CPA}$ encryption scheme. For simplicity we assume that $\mathrm{Enc}_{pk} : \mathbb{Z}_b \to \mathbb{Z}_n$, $\mathrm{Dec}_{sk} : \mathbb{Z}_n \to \mathbb{Z}_b$, and $\mathrm{Enc}_{pk}(m_1) \cdot \mathrm{Enc}_{pk}(m_2) = \mathrm{Enc}((m_1 + m_2) \bmod b)$, and the immunization function $f_{\mathsf{seed}}(r) = (\mathsf{seed} \cdot r) \bmod n$ (e.g., one can think of Benaloh cryptosystem [6]).

By $3\mathsf{rseek}$ we mean the rseek-game where the adversary gets to see 3 outputs rather than just one.

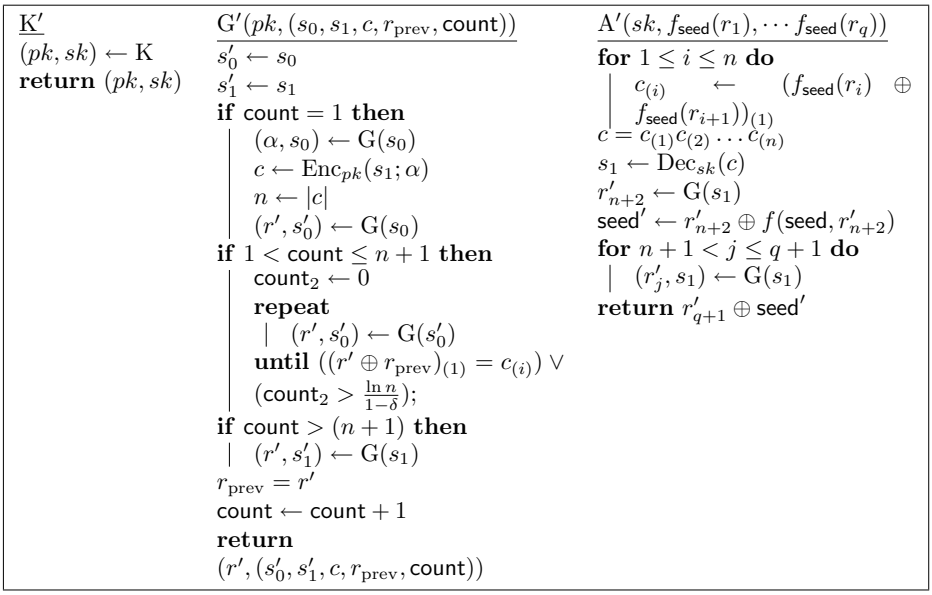

Fig. 14. Predicting backdoored PRG — Private immunization with $f_{\mathsf{seed}}(x) = \mathsf{seed} \oplus x$

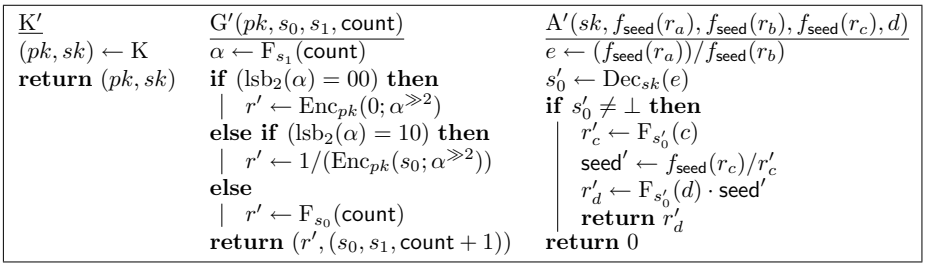

Fig. 15. Predicting Backdoored PRG — Private Immunization

Lemma 7. *Let* $(\mathrm{K}, \mathrm{Enc}_{pk}, \mathrm{Dec}_{sk})$ *be a* $(t, q, \delta) - \mathsf{IND\$\text{-}CPA}$ *public key encryption scheme which is multiplicatively homomorphic as above,* F_{sk} *be a* (t, q, δ)*-pseudorandom function for* $q \geq 4$*. Then for* $(\mathrm{K}', \mathrm{G}', \mathrm{A}')$ *defined in Fig. 15 and* $f_{\mathsf{seed}}(x) = \mathsf{seed} \cdot x$*,* $(\mathrm{K}', f_{\mathsf{seed}} \circ \mathrm{G}'(\cdot), \mathrm{A}'(\cdot))$ *is a* $(t, q, 3\delta, (\mathcal{G}_{3\mathsf{rseek}}, \frac{1}{64} - \delta))$*-backdoored pseudorandom generator.*

Proof. From pseudorandomness of F's outputs

$$\mathbf{CD}_t((\mathrm{F}_{s_0}(1), \ldots, \mathrm{F}_{s_0}(q)), \mathcal{U}) \leq \delta, \mathbf{CD}_t((\mathrm{F}_{s_1}(1), \ldots, \mathrm{F}_{s_1}(q)), \mathcal{U}) \leq \delta.$$

Then $\mathbf{CD}_t((pk, s_0, \mathrm{Enc}(s_0; \alpha^{\gg 2})), (pk, s_0, \mathcal{U})) \leq 2\delta$ due to $\mathsf{IND\$\text{-}CPA}$ security. Thus,

$$\mathbf{CD}_t((pk, \mathsf{out}^q(\mathrm{G}'_{pk}, \mathcal{U})), \mathcal{U}) \leq 3\delta.$$

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{BPRG}}_{\mathsf{3rseek}}(\mathrm{K}', f_{\mathsf{seed}} \circ \mathrm{G}', \mathrm{A}') = \Pr[r_d = r'_d] \geq \\
\Pr[\mathrm{lsb}(\mathrm{F}_{s_1}(d)) = 1 \wedge \mathsf{seed}' = \mathsf{seed} \wedge s'_0 = s_0] \geq \\
\Pr[\mathrm{lsb}(\mathrm{F}_{s_1}(d)) = 1 \wedge r'_c = r_c \wedge s'_0 = s_0] \geq \\
\Pr[\mathrm{lsb}(\mathrm{F}_{s_1}(d)) = 1 \wedge \mathrm{lsb}(\mathrm{F}_{s_1}(c)) = 1 \wedge s'_0 = s_0] \geq \\
\Pr[\mathrm{lsb}(\mathrm{F}_{s_1}(d)) = 1 \wedge \mathrm{lsb}(\mathrm{F}_{s_1}(c)) = 1 \wedge r_a = \mathrm{Enc}_{pk}(0) \wedge r_b = 1/(\mathrm{Enc}_{pk}(s_0))] \geq \\
\Pr[\mathrm{lsb}(\mathrm{F}_{s_1}(d)) = 1 \wedge \mathrm{lsb}(\mathrm{F}_{s_1}(c)) = 1 \wedge \mathrm{lsb}_2(\mathrm{F}_{s_1}(a)) = 00 \wedge \mathrm{lsb}_2(\mathrm{F}_{s_1}(b)) = 10] \geq \\
\frac{1}{64} - \delta
\end{aligned}$$

for $q \geq 4$.

□

References

1. Albertini, A., Aumasson, J.P., Eichlseder, M., Mendel, F., Schläffer, M.: Malicious hashing: Eve's variant of SHA-1. Cryptology ePrint Archive, Report 2014/694 (2014). http://eprint.iacr.org/
2. Aranha, D.F., Fouque, P.A., Qian, C., Tibouchi, M., Zapalowicz, J.C.: Binary elligator squared. Cryptology ePrint Archive, Report 2014/486 (2014). http://eprint.iacr.org/
3. Backes, M., Cachin, C.: Public-key steganography with active attacks. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 210–226. Springer, Heidelberg (2005)
4. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013)
5. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against aass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014)
6. Benaloh, J.: Dense probabilistic encryption. In: Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120–128 (1994)
7. Bendel, M.: Hackers describe PS3 security as epic fail, gain unrestricted access. http://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/
8. Bernstein, D.J., Chang, Y.-A., Cheng, C.-M., Chou, L.-P., Heninger, N., Lange, T., van Someren, N.: Factoring RSA keys from certified smart cards: coppersmith in the wild. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 341–360. Springer, Heidelberg (2013)
9. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: Elliptic-curve points indistinguishable from uniform random strings. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 967–980. ACM (2013)
10. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM Journal on Computing **15**(2), 364–383 (1986)
11. Brown, D., Vanstone, S.: Elliptic curve random number generation (2007). http://www.google.com/patents/US20070189527

12. Cachin, C.: An information-theoretic model for steganography. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 306–318. Springer, Heidelberg (1998)
13. Checkoway, S., Fredrikson, M., Niederhagen, R., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H.: On the practical exploitability of Dual EC DRBG in TLS implementations (2014)
14. Everspaugh, A., Zhai, Y., Jellinek, R., Ristenpart, T., Swift, M.: Not-so-random numbers in virtualized linux and the Whirlwind RNG (2014)
15. Goh, E.-J., Boneh, D., Pinkas, B., Golle, P.: The design and implementation ofprotocol-based hidden key recovery. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 165–179. Springer, Heidelberg (2003)
16. Goldberg, I., Wagner, D.: Randomness and the Netscape browser. Dr Dobb's Journal pp. 66–71 (1996)
17. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In: USENIX Security, pp. 205–220. USENIX (2012)
18. Holenstein, T.: Key agreement from weak bit agreement. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 664–673. ACM (2005)
19. Hopper, N., von Ahn, L., Langford, J.: Provably secure steganography. IEEE Transactions on Computers **58**(5), 662–676 (2009)
20. 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)
21. Möller, B.: A public-key encryption scheme with pseudo-random ciphertexts. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 335–351. Springer, Heidelberg (2004)
22. Mowery, K., Wei, M., Kohlbrenner, D., Shacham, H., Swanson, S.: Welcome to the Entropics: Boot-time entropy in embedded devices, pp. 589–603. IEEE (2013)
23. National Institute of Standards and Technology: Special Publication 800–90: Recommendation for random number generation using deterministic random bit generators (2012), http://csrc.nist.gov/publications/PubsSPs.html#800-90A, (first version June 2006, second version March 2007)
24. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: NDSS (2010)
25. Schoenmakers, B., Sidorenko, A.: Cryptanalysis of the dual elliptic curve pseudorandom generator. IACR Cryptology ePrint Archive **2006**, 190 (2006)
26. Shoup, V.: A proposal for an iso standard for public key encryption (version 2.1). IACR E-Print Archive 112 (2001)
27. Shumow, D., Ferguson, N.: On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. In: Proc. Crypto 2007 (2007)
28. Simmons, G.J.: The prisoners' problem and the subliminal channel. In: Advances in Cryptology. pp. 51–67. Springer (1984)
29. Tibouchi, M.: Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. Cryptology ePrint Archive, Report 2014/043 (2014). http://eprint.iacr.org/
30. Vazirani, U.V., Vazirani, V.V.: Trapdoor pseudo-random number generators, with applications to protocol design. FOCS **83**, 23–30 (1983)
31. Vazirani, U.V., Vazirani, V.V.: Efficient and secure pseudo-random number generation. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 193–202. Springer, Heidelberg (1985)

32. von Ahn, L., Hopper, N.J.: Public-key steganography. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 323–341. Springer, Heidelberg (2004)
33. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In: SIGCOMM Conference on Internet Measurement, pp. 15–27. ACM (2009)
34. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)
35. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997)
36. 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)