Cliptography: Clipping the Power of Kleptographic Attacks

Alexander Russell[1], Qiang Tang[2], Moti Yung[3], and Hong-Sheng Zhou[4]

[1] University of Connecticut, acr@cse.uconn.edu
[2] New Jersey Institute of Technology, qiang@njit.edu
[3] Snapchat Inc.& Columbia University, moti@cs.columbia.edu
[4] Virginia Commonwealth University, hszhou@vcu.edu

Abstract. Kleptography, introduced 20 years ago by Young and Yung [Crypto '96], considers the (in)security of malicious implementations (or instantiations) of standard cryptographic primitives that may embed a "backdoor" into the system. Remarkably, crippling subliminal attacks are possible even if the subverted cryptosystem produces output indistinguishable from a truly secure "reference implementation." Bellare, Paterson, and Rogaway [Crypto '14] recently initiated a formal study of such attacks on symmetric key encryption algorithms, demonstrating that kleptographic attacks can be mounted in broad generality against randomized components of cryptographic systems.

We enlarge the scope of current work on the problem by permitting adversarial subversion of (randomized) key generation; in particular, we initiate the study of cryptography in the *complete subversion model*, where *all* relevant cryptographic primitives are subject to kleptographic attacks. We construct secure one-way permutations and trapdoor one-way permutations in this "complete subversion" model, describing a general, rigorous immunization strategy to clip the power of kleptographic subversions. Our strategy can be viewed as a formal treatment of the folklore "nothing up my sleeve" wisdom in cryptographic practice. We also describe a related "split program" model that can directly inform practical deployment. We additionally apply our general immunization strategy to directly yield a backdoor-free PRG. This notably amplifies previous results of Dodis, Ganesh, Golovnev, Juels, and Ristenpart [Eurocrypt '15], which require an honestly generated random key.

We then examine two standard applications of (trapdoor) one-way permutations in this complete subversion model and construct "higher level" primitives via black-box reductions. We showcase a digital signature scheme that preserves existential unforgeability when *all* algorithms (including key generation, which was not considered to be under attack before) are subject to kleptographic attacks. Additionally, we demonstrate that the classic Blum–Micali pseudorandom generator (PRG), using an "immunized" one-way permutation, yields a backdoor-free PRG.

Alongside development of these secure primitives, we set down a hierarchy of kleptographic attack models which we use to organize past results and our new contributions; this taxonomy may be valuable for future work.

1 Introduction

Consider conventional use of a cryptographic primitive in practice, such as an encryption scheme: To encrypt a desired plaintext, one simply runs an implementation (or an instantiation with particular parameters) of the encryption algorithm obtained from a hardware or software provider. Although the underlying algorithms may be well-studied and proven secure, malicious implementations or instantiations may cleverly "backdoor" the system or directly embed sensitive information—such as the secret key—into the ciphertext in a fashion that permits recovery by the provider/manufacturer but is undetectable to other parties. Notably, *such leakage is possible even if the implementation produces "functionally and statistically clean" output that is indistinguishable from that of a faithful implementation.* While the underlying concept of kleptography was proposed by Young and Yung two decades ago [27,28], striking recent examples—including those of the Snowden revelations [20]—have reawakened the security community to the seriousness of these issues [21]. As a result, the topic has received renewed attention; see, e.g., [4,2,10,18,1]. In particular, Bellare, Paterson, and Rogaway [4] studied algorithm substitution attacks—with a focus on symmetric key encryption—and demonstrated a devastating framework for such attacks that apply in broad generality to randomized algorithms. These results were later amplified [3] to show that such attacks can be carried out even if the adversarial implementation is *stateless*. Soon after, Dodis, Ganesh, Golovnev, Juels, and Ristenpart [10] formalized the subversion of Dual_EC pseudorandom generators (PRG) and studied backdoored PRGs in generality; they additionally studied methods for "immunizing" PRG in such hostile settings.

Our contributions. We continue this line of inquiry. Specifically, we are motivated to develop cryptographic schemes in a *complete subversion model*, in which *all* algorithms of a scheme are potentially subverted by the adversary. This model provides a conceptually simple abstraction of the adversary's power, and significantly amplifies previously studied settings, which rely on trusted key generation or clean randomness that is assumed private from the adversary.

In particular, motivated by the question of defending against the kleptographic attacks on **key generation** as demonstrated in the original paper of [27,28], we study two fundamental cryptographic primitives in the complete subversion model—one-way permutations (OWP) and trapdoor one-way permutations (TOWP)—and apply these primitives to construct other cryptographic tools such as digital signatures and PRGs. Along the way, we identify novel generic defending strategies and a hierarchy of attack models. We hope to stimulate a systematic study of "**cliptography**," the challenge of developing a broad class of familiar cryptograhpic tools that remain secure in such kleptographic settings. As mentioned above, prior to our work kleptographic attacks on various primitives have been addressed in weaker models; see the discussion of related work in Section 1. In detail, we show the following:

- We set down a hierarchy of security models that capture practical kleptographic settings. The models are characterized by three parties: an adversary, who may provide potentially subverted implementations of *all* cryptographic algorithms; a "watchdog," who either certifies or rejects the implementations by subjecting them to (black-box) interrogation;[5] and a challenger, who plays a conventional security game (but now using the potentially subverted algorithms) with the adversary. Armed with the "specification" of the cryptographic algorithms and oracle access to the implementations provided by the adversary, the watchdog attempts to detect any subversion in the implementations. Various models arise by adjusting the supervisory power of the watchdog; see Section 2.

- We study (trapdoor) one-way permutations in the presence of kleptographic attacks, introducing notions of subversion-resistance that can survive various natural kleptographic attacks. We first give a simple example of a OWP that can be proven secure in the conventional sense, but can be completely broken under the kleptograhic attack. This demonstrates the need for judicious design of cryptographic primitives to defend against kleptographic attacks.

  We then construct subversion-resistant (trapdoor) one way permutations via a general transformation that "sanitizes" arbitrary OWPs by *randomizing the function index*. This transformation clips potential correlation between the function and the possible backdoor that the adversary may possess. Additionally, we introduce a *split-program* model to make the general method above applicable using standard hash functions (see Section 3.3).
- In Section 4, we observe that subversion-resistant trapdoor OWPs give us a way to construct key generation algorithms (for digital signature schemes) against kleptographic attacks. We then showcase a concrete example of a digital signature scheme in the complete subversion model. More concretely, we achieve this result by (1) using the subversion-resistant trapdoor OWP directly as a key generation algorithm, and then (2) instantiating the unique signature generation mechanism via full domain hash (FDH). We stress that the reduction of the standard FDH signature scheme does not go through in the kleptographic setting. To resolve this issue, we slightly modify the FDH approach by hashing the message *together with the public key*. We remark that the original kleptographic attacks [27,28] were indeed applied to the key generation algorithm, while recent efforts [4,1] shift focus to other algorithmic aspects of encryption or digital signature schemes, assuming that key generation is honest. Our result is the first digital signature scheme allowing the adversary to sabotage all algorithms, including key generation.
- We then turn our attention to PRGs. Previous work of Dodis et al. [10] investigated a notion of "backdoored PRG" in which the adversary sets up a PRG instance (i.e., the public parameter), and is able to distinguish the output from uniform with a backdoor. They then proposed powerful immunizing strategies which apply a keyed hash function to the output—*assuming the key is unknown to the adversary*—in the public parameter generation phase. Motivated by their success, we focus on constructing backdoor-free

[5] Without the watchdog, it is elusive to achieve interesting cryptographic functionalities in those stringent settings.

PRGs in the complete subversion model (where such clean randomness is not permitted). Our first construction is based on the classic Blum-Micali construction, using our subversion-resistant OWP and the Goldreich-Levin hardcore predicate. Dodis et al. [10] additionally show that it is impossible to achieve a public immunizing strategy for all PRGs by applying a public function to the PRG output. We sidestep this impossibility result via an alternative public immunizing strategy: Rather than randomizing the output of the PRG, we randomize the public parameter of PRG, which yields a general construction for PRG in the complete subversion model. See Section 5.

Finally, we remark that black-box constructions and reductions do not, in general, survive in the kleptographic model. However, two of the results above—the Blum-Micali construction and the signature scheme—give explicit examples of reductions that can be salvaged.

Remarks: Our techniques and the "nothing up my sleeve" principle; single use of randomized algorithms and subliminal channels. We remark that our general defending technique significantly differs from known methods: We use a—potentially subverted—hash function to "randomize" the index and public parameter of a (perhaps randomized) algorithm so that any correlation with some potential backdoor can be eliminated. This can be seen as an instance of the folklore wisdom of a "nothing up my sleeve number" [26] which has been widely used in practical cryptographic designs. The basic principle calls for constants appearing in the development of cryptographic algorithms to be drawn from a "rigid" source, like the digits of π; the idea is that this prevents them from possessing hidden properties that might give advantage to an attacker (or the designer). In our setting, the fact that a given value v is *supplied along with a preimage x so that $h(x) = v$* (for a hash function h) is a evidence that v has "nothing up its sleeve." In fact, the situation is complicated: While this does effectively mean that v is generated by selecting x and computing $h(x)$ and, thus, severely restricts the possibility for tampering with v, it does not eliminate subliminal channels introduced by rejection sampling or entirely "clean" v. In particular, detailed analysis is still required to control the behavior of v.

Previous results either use a trusted random source to re-randomize the output of a randomized algorithm, or consider only deterministic algorithms. Permitting randomized algorithms in a kleptographic framework immediately invites the (devastating) general "steganochannel" attack of Bellare et al. [4,3]. Apparently, the prospect of full "immunization" for general randomized algorithms (in particular, generic destruction of a steganochannel) is a presumably challenging direction of future work. We note that our primitives here do permit randomized algorithms, although the security games we analyze invoke them only once (to, e.g., derive a key). Very interestingly, recent subsequent work of Russell et al [23] addresses this major problem for a class of randomized algorithms; as a consequence, they can achieve the first IND-CPA secure public key encryption in the kleptographic setting.

For simplicity, we focus on (potentially subverted) algorithms that do not maintain "internal state" between invocations. We remark that typical steganographic attacks can indeed be in carried out in a stateless model [3]. Moreover, this restriction can be lifted for the constructions in the paper; see Remark 1.

Related work. The concept of *kleptography*—subverting cryptographic algorithms by modifying their implementations to leak secrets covertly—was proposed by Young and Yung [27,28] in 1996. They gave concrete examples showing that backdoors can be embedded into the public keys of commonly used cryptographic schemes; while the resulting public keys appear normal to every user, the adversary is nevertheless capable of learning the secret keys. Young and Yung have shown kleptographic backdoors in digital signature algorithms, key exchanges, SSL, symmetric crypto (e.g., block ciphers), composite key generation (e.g., RSA), and public key cryptosystems [27,28,29,30,31,32]. It may not be surprising that defending against such deliberate attacks is challenging and only limited feasibility results exist. We next briefly describe these existing results.

In [16], Juels and Guajardo suggested the following idea: the user and a trusted certificate authority (CA) jointly generate the public key; as a part of this process, the user proves to the CA that the public key is generated honestly. This contrasts markedly with our setting, where the the user does not have any secret, and every component is provided by the adversary.

Bellare et al. considered a powerful family of kleptographic attacks that they call *algorithm substitution attacks*, and explore these in both symmetric-key [4,3] and public-key [2] settings. They first proposed a generic attack, highlighting the relevance of steganographic techniques in this framework: specifically, a sabotaged randomized algorithm can leak a secret bit-by-bit by invoking steganographic rejection-sampling; then an adversary possessing the backdoor can identify the leaked bits from the biased output, which appears unmolested to other observers. The attack and analysis relies on the effectiveness of subliminal channels [24,25,15]. They then introduced a framework for defending against such attacks by focusing on algorithms that having a unique output for each input: relevant examples of such algorithms include unique ciphertext encryption algorithms. These results were later refined by [9]. Their defending mechanism does not, however, address the (necessarily randomized) process of key generation—it implicitly assumes key generation to be honest. This state of affairs is the direct motivation of the current article: we adopt a significantly amplified *complete subversion model* where *all* cryptographic algorithms—including key generation—are subject to kleptographic (i.e., substitution) attacks. The details of the model, with associated commentary about its relevance to practice, appear below.

Dodis et al. [10] pioneered the rigorous study of pseudorandom generators in such settings, developing an alternative family of kleptographic attacks on pseudorandom generators in order to formalize the notorious Dual_EC PRG subversion [19,7]. In their model, the adversary subverts the security of the PRG by opportunistically setting the public parameter while privately keeping some backdoor information (instead of providing an implementation). They then demonstrate an impossibility result: backdoored PRGs cannot be immunized by applying a public function—even a trusted random oracle—to the output. They also proposed and analyzed immunizing strategies obtained by applying a keyed hash function to the output (of the PRG). Note that the (hash) key plays a special role in their model: it is selected uniformly and is unknown to the adversary during the public parameter generation phase. These results likewise inspire our adoption of the amplified *complete subversion model*, which excludes such reliance on public randomness beyond the reach of the adversary. In particular, our general immunizing strategy (randomizing the public parameter of a backdoored PRG instead of randomizing the PRG output) permits us to bypass the impossibility result. Additionally, our results on subversion-resistant OWFs can be applied to construct a specific "backdoor-free" PRG following the classic Blum-Micali framework.

Other works suggest different angles of defense against mass surveillance. For example, in [18,11] the authors proposed a general framework of safeguarding protocols by randomizing the incoming/outgoing messages via a trusted (reverse) firewall. Their results demonstrate that with a trusted random source, many tasks become achievable. As they rely on a "subversion-free" firewall, these results require a more generous setting than provided by our *complete subversion model*.

Ateniese et al. [1] continued the study of algorithm substitution attacks on signatures and propose two defending mechanisms, one utilizes a unique signature scheme assuming the key generation and verify algorithms to be honest; the other adopts the reverse firewall model that assumes trusted randomness. We construct a signature scheme that can be proven secure in the complete subversion model which does not make assumptions on honesty or require trusted randomness. We remark that the strength of the "watchdog" that is required for the signature scheme is, however, stronger than that required for the other primitives; it must be permitted a transcript of the security game. See Section 4.

2 A Definitional Framework for Cliptography

2.1 From Cryptography to Cliptography

In this section, we lay down a definitional framework for cliptography. The adversary in this new setting is "*proud-but-malicious*": the adversary wishes to supply a subverted implementation in order to break security while keeping the subversion "under the radar" of any detection. Thus the basic framework should reflect the ability of the adversary to provide (potentially subverted) implementations of the cryptographic algorithms of interest, the ability of an efficient "watchdog" to interrogate such implementations in order to check their veracity, and a classical "challenger-adversary" security game. Specifically, the model considers an adversary that commences activities by supplying a (potentially subverted) implementation of the cryptographic primitive; one then considers two parallel procedures: a classical challenger-adversary security game in which the challenger must use only (oracle access to) the adversary's implementations, and a process in which the "watchdog" compares—also via oracle access—the adversary's implementations against a specification of the primitives. (For entertainment, we occasionally refer to the adversary as "big brother.")

Cryptographic games. We express the security of (standard) cryptographic schemes via *cryptographic games* between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Definition 1 (Cryptographic Game [14]). *A cryptographic game* $\mathsf{G} = (\mathcal{C}, \delta)$ *is defined by a random system* $\mathcal{C}$*, called the challenger, and a constant* $\delta \in [0, 1)$*. On security parameter* λ*, the challenger* $\mathcal{C}(1^\lambda)$ *interacts with some adversary* $\mathcal{A}(1^\lambda)$ *and outputs a bit* b*. We denote this interaction by* $b = (\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}(1^\lambda))$*. The advantage of an attacker* $\mathcal{A}$ *in the game* G *is defined as*

$$\mathbf{Adv}_{\mathcal{A},\mathsf{G}}(1^\lambda) = \Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}(1^\lambda)) = 1\right] - \delta\,.$$

We say a cryptographic game G *is* secure *if for all* PPT *attackers* $\mathcal{A}$*, the advantage* $\mathbf{Adv}_{\mathcal{A},\mathsf{G}}(1^\lambda)$ *is negligible in* λ*.*

The above conventional security notions are formulated under the assumption that the relevant algorithms of the cryptographic scheme are faithfully implemented and, moreover, that participants of the task have access to truly private randomness (thus have, e.g., truly random keys). In the kleptographic setting, these assumptions are relaxed.

The complete subversion model. A basic question that must be addressed by a kleptographic model concerns the selection of algorithms the adversary is permitted to subvert. We work exclusively in a setting where the adversary is permitted to provide implementations of *all* the relevant cryptographic elements of a scheme, a setting we refer as the *complete subversion* model. Thus, all guarantees about the quality of the algorithms are delivered by the watchdog's testing activities. This contrasts with all previous work, which explicitly protected some of the algorithms from subversion, or assumed clean randomness. Such a setting we refer to as *partial subversion* model.

Choosing the right watchdog. By varying the information provided to the watchdog, one obtains different models that reflect various settings of practical interest. The weakest (and perhaps most attractive) model is the *offline* watchdog, which simply interrogates the supplied implementations, comparing them with the specification of the primitives, and declares them to be "fit" or "unfit." Of course, we must insist that such watchdogs find the actual specification "fit": formally, the definition is formulated in terms of distinguishing an adversarial implementation from the specification.

One can strengthen the watchdog by permitting it access to the full transcript of the challenger-adversary security game, resulting in the *online* watchdog. Finally, we describe an even more powerful *omniscient* watchdog, which is even privy to private state of the challenger. (While we do not use such a powerful watchdog in our results, it is convenient for discussing previous work.)

We remark these various watchdogs reflect various levels of "checking" that a society might entertain for cryptographic algorithms (and conversely, various levels of tolerance that an adversary may have to exposure): the offline watchdog reflects a "one-time" laboratory that attempts to check the implementations; an online watchdog actually crawls public transcripts of cryptographic protocols to detect errors; the omniscient watchdog requires even more, involving (at least) individuals effectively checking their results against the specification.

2.2 A Formal Definition

Having specified the power of the big brother (the adversary) and that of the watchdog, we are ready to introduce *cliptographic games* to formulate security. To simplify the presentation, we here initially consider *complete* subversion with an *offline* watchdog. In the next section, we will discuss the other variants.

A cryptographic scheme $\mathit{\Pi}$ consists of a set of (possibly randomized) algorithms $(F^1, \ldots, F^k)$. (In general, deterministic algorithms determine functions $F^i : (\lambda, x) \mapsto y$, whereas randomized algorithms determine distributions $F^i(\lambda, x)$ over an output set Y_λ.) For example, a digital signature scheme consists of three algorithms, a (randomized) key generation algorithm, a signing algorithm, and deterministic verification algorithm. The definition of $\mathit{\Pi}$ results in a *specification* of the associated algorithms; for concreteness, we label these as $\mathit{\Pi}_{\text{SPEC}} = (F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}})$; when a scheme is (perhaps adversarially) implemented, we denote the implementation as $\mathit{\Pi}_{\text{IMPL}} = (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$. If the implementation honestly follows the specification of the scheme, we overload the notation and represent them interchangeably with the specification as $\mathit{\Pi}_{\text{SPEC}}$.

In our definition, the adversary $\mathcal{A}$ will interact with both the challenger $\mathcal{C}$ and the watchdog $\mathcal{W}$. (In the offline case, these interactions are independent; in the online case, $\mathcal{W}$ is provided a transcript of the interaction with $\mathcal{C}$.) Following the definition of cryptographic game, we use $b_{\mathcal{C}} = (\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}}}(1^\lambda))$ to denote the interaction between $\mathcal{A}$ and $\mathcal{C}$; $b_{\mathcal{C}}$ denotes the bit returned by the challenger $\mathcal{C}$. (Note that the challenger must use the implementation of $\mathit{\Pi}$ provided by the adversary, while the interaction between $\mathcal{A}, \mathcal{C}$ is the same as in the classical cryptographic game.)

As for the watchdog $\mathcal{W}$, the adversary provides $\mathcal{W}$ his potentially subverted implementations $\mathit{\Pi}_{\text{IMPL}}$ of the primitive (as oracles); $\mathcal{W}$ may then interrogate them in an attempt to detect divergence from the specification, which he possesses. On the basis of these tests, the watchdog produces a bit (Intuitively, the bit indicates whether the implementations passed whatever tests the watchdog carried out to detect inconsistencies with the specification.)

Definition 2 (Cliptographic Game). *A cliptographic game $\widehat{\mathsf{G}} = (\mathcal{C}, \mathit{\Pi}_{\text{SPEC}}, \delta)$ is defined by a challenger $\mathcal{C}$, a specification $\mathit{\Pi}_{\text{SPEC}}$, and a constant $\delta \in [0, 1)$. Given an adversary $\mathcal{A}$, a watchdog $\mathcal{W}$, and a security parameter λ, we define the* detection probability *of the watchdog $\mathcal{W}$ with respect to $\mathcal{A}$ to be*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left| \Pr[\mathcal{W}^{F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}}}(1^\lambda) = 1] \right| ,$$

where $\mathit{\Pi}_{\text{IMPL}} = (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$ denotes the implementation produced by $\mathcal{A}$. The advantage *of the adversary is defined to be*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \left| \Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}}}(1^\lambda)) = 1 \right] - \delta \right| .$$

We say that a game is ***subversion-resistant*** *if for any polynomial $q(\cdot)$, there exists a* PPT *watchdog $\mathcal{W}$ such that for all* PPT *adversaries $\mathcal{A}$, either $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda)$ is non-negligible, or $\mathbf{Adv}_{\mathcal{A}}(1^\lambda)$ is negligible, in the security parameter λ.*

Other watchdog variants. In the above definition, we chose the strongest model: the watchdog is *universal* and *offline*. In particular, primitives secure in this model are secure in any of the other models considered. The detection algorithm of the watchdog must be designed for a *given specification*, regardless of how the adversary subverts the implementation; furthermore, it may only carry out a one-time check on the implementation (and may not supervise the security game). To permit a broader class of feasibility results, it is possible to extend the basic model in both directions.

Swapping the quantifiers. It is also reasonable to consider a watchdog that may be tailored to the adversary, i.e., the quantifiers are changed to be $\forall \mathcal{A}, \exists \mathcal{W}$. Indeed, such quantification (or even more generous settings, see below) was considered implicitly in previous works, e.g., [4,10,3]. We remark that such a model is still highly non-trivial in that the adversary can be randomized by, e.g., selecting a random backdoor. (Thus knowing the code of the adversary does not necessarily help the watchdog to identify a faulty implementation which might be based on a random backdoor that is only known to $\mathcal{A}$.) Note that such a model is particularly interesting for evaluating *attacks*, where one would like to guarantee that the attack is undetectable even by a watchdog privy to the details of the adversary: specifically, when establishing security, weak watchdogs are preferable; when establishing the value of an attack, strong watchdogs are preferable.

We develop one-way permutations and pseudorandom generators in the offline model. However, it appears that richer primitives may require qualitatively stronger watchdogs. Considering that an offline watchdog cannot ensure *exact* equality for deterministic algorithms, we remark that a clever adversary may be able to launch attacks by altering a deterministic function at a single location. Imagine a security game where the adversary supplies a string m to which the challenger is expected to apply one of the subverted algorithms; this takes place, e.g., in the typical signature security game. The adversary may now select a random string w and implement the deterministic algorithm in such a way that it diverges from the specification at (only) this preselected point. While such inconsistencies are (essentially) undetectable by an offline watchdog, the adversary can ensure that the subverted algorithm is indeed queried at w during the security game. Such "input-triggering attacks" in [4,9,1] motivated them to consider extra "decryptability condition" and "verifiability condition" assumptions.

An *online watchdog* can guard against this possibility; he is permitted to monitor the public interactions between users. More precisely, the online watchdog is permitted to certify both the implementations *and* the transcript between the challenger and adversary. The security game is then altered by considering $\mathcal{W}^{\Pi_{\text{IMPL}}}(1^\lambda, \tau)$, identical to the offline case except that the watchdog is provided the transcript τ of the security game $(\mathcal{C} \Leftrightarrow \mathcal{A})$.[6] (We use the shorthand notation Π_{IMPL} here to denote the collection of oracles $F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}}$.) The detection game must then be adjusted, guaranteeing that the transcripts produced when the challenger uses Π_{IMPL} are indistinguishable from those produced when the challenger uses Π_{SPEC}. Our results on digital signature schemes will require

[6] We remark that the transcript τ includes the final output bit of the challenger.

such a watchdog. We remark that previous work on subversion-resistant digital signatures [1] assumes a verifiability condition: every message-signature pair produced by the subverted sign algorithm (at least the responses to the signing queries) can pass the verification of the specification of the verify algorithm. This extra assumption can be guaranteed by an online watchdog (and, indeed, it demands either an absolute universal guarantee or an on-line guarantee for those pairs that appear in the security game).

An *omniscient watchdog* is even stronger. In addition to access to the transcript, the omniscient watchdog is aware of the entire internal state of the challenger (and can monitor the interactions between users and the subverted implementations). Similarly, by replacing $\mathcal{W}$ in Definition 2 above with an omniscient watchdog, we obtain cliptographic games with omniscient watchdog. As mentioned, omniscient watchdog has been considered in literature [4,9]. In those works, they assume the extra decryptability condition such that ciphertext generated by the subverted encryption algorithm decrypts correctly with the honest decryption algorithm. Again, without allowing the watchdog to input the whole transcript and the decryption key, this assumption cannot supported.

Discussion: The guarantees provided by an offline watchdog. We make some general observations about the guarantees that an offline watchdog provides.

Consider a *deterministic* algorithm implemented by the adversary; an offline watchdog cannot ensure that such an algorithm is perfectly implemented. However, it can ensure that the implementation agrees with the specification with high probability over a particular (sampleable) distribution of choice (by simply drawing from the distribution and checking equality). This frequently arises in our setting, where we are led to study the behavior of a deterministic algorithm on a particular "public input distribution."

Lemma 1. *Consider an adversarial implementation $\Pi_{\text{IMPL}} := (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$ of a specification $\Pi_{\text{SPEC}} = (F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}})$ in a cliptographic game, where $F^1, \ldots, F^k$ are deterministic algorithms. Additionally, for each security parameter λ, (sampleable) public input distributions $X^1_\lambda, \ldots, X^k_\lambda$ are defined respectively. If $\exists j \in [k],\ \Pr[F^j_{\text{IMPL}}(x) \neq F^j_{\text{SPEC}}(x) : x \leftarrow X^j_\lambda]$ is non-negligible, then there is a* PPT *offline watchdog that can detect with a non-negligible probability.*

The above includes the cases that the deterministic algorithm is with a known input distribution (e.g, uniform distribution), or with an input distribution that is generated by other (adversarial) implementations. Jumping ahead, the evaluation function of a one way permutation takes a uniform input distribution; and a pseudorandom generator stretch function takes $\mathcal{K} \times \mathcal{U}$ as (public) input distribution, where $\mathcal{K}$ is the output distribution of a parameter generation algorithm implemented by the adversary and $\mathcal{U}$ is the uniform seed distribution.

In our analysis, we will use this simple observation extensively. In particular, when a hash specification is modeled as a random oracle we can check that the hash function has been faithfully implemented by the adversary (with high probability) for any particular sampleable distribution of choice; in many cases, these will be distributions generated by other adversarial implemented algorithms.

Next, consider a *randomized* algorithm (with fixed inputs) that is supposed to output a high-entropy distribution. The offline watchdog can provide a weak guarantee of min-entropy by simply running the algorithm twice to see whether there is collision. While this does not guarantee large entropy, it can guarantee a critical feature: the result is unpredictable to the adversary.

Lemma 2. *Consider an adversary $\mathcal{A}$ which prepares the implementation F_{IMPL} of a specification F_{SPEC}, where F_{SPEC} is a randomized algorithm that produces an output distribution with $\omega(\log\lambda)$ min-entropy. If $\Pr[x = x' : x \leftarrow \mathcal{A}(\lambda), x' \leftarrow F_{\text{IMPL}}] \leq \mathsf{negl}(\lambda)$ does not hold, then there is a* PPT *offline watchdog that can detect this with a non-negligible probability.*

Discussion: random oracles. In many settings, we establish results in the conventional *random oracle* model which requires some special treatment in the model above. In general, we consider a random oracle to be an (extremely powerful) heuristic substitute for a deterministic function with strong cryptographic properties. In a kleptographic setting with complete subversion, we must explicitly permit the adversary to tamper with the "implementation" of the random oracle supplied to the challenger. In such settings, we provide the watchdog—as usual—oracle access to both the "specification" of the random oracle (simply a random function) and the adversary's "implementation" of the random oracle, which may deviate from the random oracle itself. For concreteness, we permit the adversary to "tamper" with a random oracle h by providing an efficient algorithm $T^h(x)$ (with oracle access to the random oracle h) which computes the "implementation" $\tilde{h}$—thus the implementation $\tilde{h}(x)$ is given by $T^h(x)$ for all x. Likewise, during the security game, the challenger is provided oracle access only to the potentially subverted implementation $\tilde{h}$ of the random oracle. As usual, the probabilities defining the security (and detection) games are taken over the choice of the random oracle. In this sense, the random oracle assumption used in our complete subversion model is weaker than the classical one, since we can allow even "imperfect" random oracles. Fortunately, when the random oracle is applied to a known input distribution, an offline watchdog can ensure that the implementation is almost consistent with its specification (see Lemma 1).

Remark 1. **Stateless/stateful implementations.** In principle, algorithms in the specification of a cryptographic scheme or implementations provided by an adversary could be stateful; for simplicity, we focus on stateless implementations in the above lemmas. However, to jump ahead a bit, those results still hold (with simple modifications) in natural stateful settings. To see this, (1) consider a randomized algorithm specified to produce a high-entropy output distribution: in the case of a stateful implementation (maintaining a local state), the unpredictability requirement can still be ensured by an offline watchdog who can *rewind* the implementation. The watchdog simply samples (rewinds to the same state and then samples) from the randomized algorithm to see whether there is a collision. (2) For deterministic algorithms with a state, as an example, we consider a stateful PRG, where the seed is updated in each iteration. In this case, the public input distribution is evolving during the iterations. Observe that the offline watchdog can indeed ensure the consistency of the implementation of the PRG when the input is chosen from a uniform distribution. This means the "bad" input set (on which the implementation deviates from its specification) could be at most negligibly small (in the uniform distribution). Note that starting from a uniform seed, any polynomially number of PRG iterations will yield poly-many pseudorandom bits. Thus the probability for any of them falls into the "bad" input set would still be negligible.

Schemes with augmented system parameter. Often, deployment of a cryptographic scheme may involve a *system parameter generation* algorithm $pp \leftarrow \mathsf{Gen}(1^\lambda)$. When we consider such an augmented scheme $\Pi = (\mathsf{Gen}, F^1, F^2, F^3)$ in our setting, we can treat the system parameter pp in two natural ways: (1.) as in Definition 2, the adversary simply provides the implementation $\mathsf{Gen}_{\text{IMPL}}$ to $\mathcal{W}$ (and $\mathcal{C}$) as usual and the challenger computes pp by running $\mathsf{Gen}_{\text{IMPL}}$ during the security game; (2) the *adversary provides* pp directly to the watchdog $\mathcal{W}$ (and $\mathcal{C}$); we write $\mathcal{W}^{\Pi_{\text{IMPL}}}(1^\lambda, pp)$ to reflect this. By replacing $\mathcal{W}^{\Pi_{\text{IMPL}}}(1^\lambda)$ in Definition 2 with $\mathcal{W}^{\Pi_{\text{IMPL}}}(1^\lambda, pp)$, and suitably changing the security game so that the challenger does not generate pp, we can obtain the *adversarially chosen parameter model.* This model was used for studying pseudorandom generator under subversion in [10], we choose to present it as a general model that would be interesting to consider for any cryptographic primitive.

It is clear that if a primitive is secure in the adversarially chosen parameter model, then it is secure according to Definition 2. (Observe that the adversary is always free to generate pp according to the algorithm provided to the challenger.) We record this below.

Lemma 3. *If Π is secure in the adversarially chosen parameter model, then Π is secure according to Definition 2.*

Schemes with split-program. Randomized algorithms play a distinguished role in our kleptographic setting. One technique we propose for immunization may also rely on the decomposition of a randomized generation algorithm $y \leftarrow \mathsf{Gen}(1^\lambda)$ into two algorithms: a random string generation algorithm RG responsible for producing a uniform poly(λ)-bit random string r, and a deterministic output generation algorithm dKG that transforms the randomness r into an output y. Note that dKG is deterministic and is always applied to a public input distribution. In light of Lemma 1, we may assume that the maliciously implemented $\mathsf{dKG}_{\text{IMPL}}$ is consistent with the honest implementation $\mathsf{dKG}_{\text{SPEC}}$ with overwhelming probability. See results in this model in Section 3.3, and definition in the full version [22].

We remark that this perspective only requires a change in the specification of Π_{SPEC}. When we apply Definition 2 with a specification that has been altered this way, we say that a primitive is proven secure in the *split-program model.*

The split-program model is quite general and can be applied to most practical algorithms. To see this, the user is provided the source code of the implementation which makes calls to some API for generating randomness (e.g., `rand()`) whenever necessary. The user can hook up the calls to the randomness API with the separate program RG provided by the adversary. (In fact, full source code is not strictly necessary in this setting; object code that adopts a particular fixed convention to gather randomness would also suffice.)

3 Subversion-Resistant One-Way Permutations

In this section, we study one-way permutations (OWP) in our cliptographic framework. As mentioned before, this is motivated by the problem of defending against subverted key generation. In particular, we propose general constructions for subversion-resistant OWPs that require only the weakest (offline) watchdog with adversarially chosen parameters. Our "immunizing strategy" consists of coupling the function generation algorithm with a hash function that is applied to the function index—intuitively, this makes it challenging for an adversary to meaningfully embed a backdoor in the permutation or its index.[7] We prove that if the specification of the hash function is modeled as a random oracle, then randomizing the permutation index using the (adversarially implemented) hash function destroys any potential backdoor structure. We emphasize that the permutation evaluation algorithm, the name generation algorithm, and the hash function may all be subverted by the adversary.

The cliptographic model introduces a number of new perspectives on the (basic) notion of security for one-way permutations. We actually consider three different notions below, each of which corresponds to distinct practical settings: the first corresponds to the classical notion, where the challenger chooses the index of the function (using subverted code provided by the adversary)—we call this $\mathrm{OWP}^{\mathrm{C}}$; the second corresponds to a setting where the adversary may choose the index—we call this $\mathrm{OWP}^{\mathrm{A}}$; the last corresponds to our "split program model," discussed above—we call this $\mathrm{OWP}^{\mathrm{SP}}$.

In many cases of practical interest, however, the permutation index may have special algebraic structure, e.g., RSA or DLP. In such cases, it would appear that the public hash function would require some further "structure preserving" property (so that it carries the space of indices to the space of indices). Alternatively, one can assume that the space of indices can be "uniformized," that is, placed in one-to-one correspondence with strings of a particular length. In order to apply our approach to broader practical settings, we apply the "split-program" model discussed above. This effectively "uniformizes" index space by insisting that the function generation algorithm is necessarily composed of two parts: a random string generation algorithm RG that outputs random bits r, and a deterministic function index generation algorithm dKG which uses r to generate the index. Hashing is then carried out on r; see Section 3.3 for details.

3.1 Defining subversion resistant OWP/TDOWP

In this subsection, following our general definitional framework, we define the security of one-way permutations and trapdoor one-way permutations. We first recall the conventional definitions.

[7] In concrete constructions, the hash function becomes a component of, e.g., the evaluation function, so that the syntax of the primitive is still the same.

One-way permutation (OWP). A family of permutations $\mathcal{F} = \{f_i : X_i \to X_i\}_{i \in I}$ is *one-way* if there are PPT algorithms $(\mathsf{KG}, \mathsf{Eval})$ so that (i) KG, given a security parameter λ, outputs a function index i from $I_\lambda = I \cap \{0,1\}^\lambda$; (ii) for $x \in X_i$, $\mathsf{Eval}(i, x) = f_i(x)$; (iii) $\mathcal{F}$ is one-way; that is, for any PPT algorithm $\mathcal{A}$, it holds that $\Pr[\mathcal{A}(i, y) \in f_i^{-1}(y) \mid i \leftarrow \mathsf{KG}(\lambda); x \leftarrow X_i; y := f_i(x)] \leq \mathsf{negl}(\lambda)$.

Trapdoor one-way permutation (TDOWP). A family of permutations $\mathcal{F} = \{f_i : X_i \to X_i\}_{i \in I}$ is *trapdoor one-way* if there are PPT algorithms $(\mathsf{KG}, \mathsf{Eval}, \mathsf{Inv})$ such that (i) KG, given a security parameter λ, outputs a function index and the corresponding trapdoor pair (i, t_i) from $I_\lambda \times T$, where $I_\lambda = I \cap \{0,1\}^\lambda$, and T is the space of trapdoors; (ii) $\mathsf{Eval}(i, x) = f_i(x)$ for $x \in X_i$; (iii) $\mathcal{F}$ is one-way; and (iv) it holds that $\Pr[\mathsf{Inv}(t_i, i, y) = x \mid i \leftarrow \mathsf{KG}(\lambda); x \leftarrow X_i; y := f_i(x)] \geq 1 - \mathsf{negl}(\lambda)$.

Sometimes, we simply write $f_i(x)$ rather than $\mathsf{Eval}(i, x)$.

Subversion-resistant$^{\mathbf{C}}$ one-way permutations: OWP$^{\mathbf{C}}$. As described in Section 2, we assume a "laboratory specification" of the OWP, $(\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}})$, which has been rigorously analyzed and certified (e.g., by the experts in the cryptography community). The adversary provides an alternate (perhaps subverted) implementation $(\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}})$. We study OWP/TDOWP in the offline watchdog model; while the implementations may contain arbitrary backdoors or other malicious features, they can not maintain any state.

Intuitively, the goal of the adversary is to privately maintain some "backdoor information" z so that the subverted implementation $\mathsf{KG}_{\text{IMPL}}$ will output functions that can be inverted using z. In addition, to avoid detection by the watchdog, the adversary must ensure that implementations $(\mathsf{KG}_{\text{IMPL}}(z), \mathsf{Eval}_{\text{IMPL}}(z))$ are computationally indistinguishable from the specification $(\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}})$ given only oracle access. Formally,

Definition 3. *A one-way permutation family $\mathcal{F} = \{f_i : X_i \to X_i\}_{i \in I}$ with the specification $\mathcal{F}_{\text{SPEC}} = (\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}})$, is* ***subversion-resistant***$^{\text{C}}$ *in the offline watchdog model if there exists a* PPT *watchdog $\mathcal{W}$, s.t.: for any* PPT *adversary $\mathcal{A}$ playing with the challenger $\mathcal{C}$ in the following game, (Fig. 1), either the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is non-negligible, or the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible.*
Here the detection probability *of the watchdog $\mathcal{W}$ with respect to $\mathcal{A}$ is defined as*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}}}(1^\lambda) = 1]\right| ,$$

and the advantage *of the adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}}(1^\lambda)) = 1\right] .$$

For convenience, we also say that such $\mathcal{F}_{\text{SPEC}}$ is a OWP^C in the offline watchdog model. On the other hand, we say that an OWP is subvertible *if:*

$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda)$ *is negligible for all* PPT *$\mathcal{W}$, and $\mathbf{Adv}_{\mathcal{A}}(1^\lambda)$ is non-negligible.* [8]

[8] We choose a stronger definition for subvertibility (swap the quantifiers of $\mathcal{A}, \mathcal{W}$) to describe attacks instead of directly negating the definition of OWP$^{\text{C}}$.

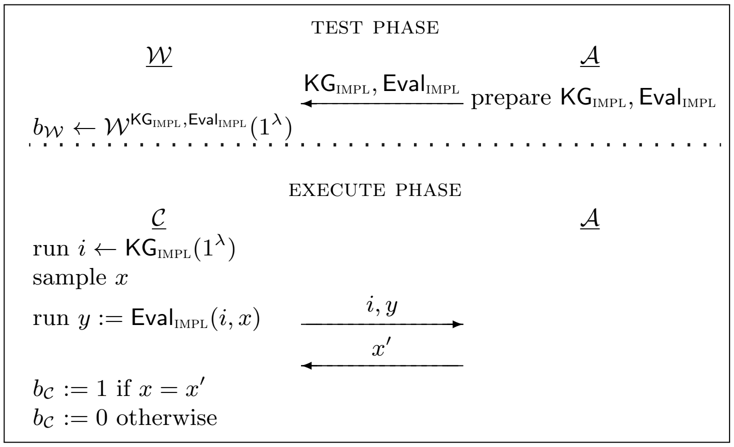

TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}$
prepare $\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}}(1^\lambda)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
run $i \leftarrow \mathsf{KG}_{\text{IMPL}}(1^\lambda)$
sample x
run $y := \mathsf{Eval}_{\text{IMPL}}(i, x)$
i, y
x'
$b_{\mathcal{C}} := 1$ if $x = x'$
$b_{\mathcal{C}} := 0$ otherwise

Fig. 1. Subversion-resistant$^{\text{C}}$ security game: OWP$^{\text{C}}$.

Subvertible OWPs. Next we observe that it is easy for an adversary to break the security of a conventional OWP in the kleptographic setting. In particular, the following lemma shows that one can construct a *subvertible* OWP (so the subverted implementation can evade detection by all watchdogs and the adversary can invert) using a conventional trapdoor OWP. In particular, if we wish to use public-key cryptography in a kleptographic setting, nontrivial effort is required to maintain the security of even the most fundamental cryptographic primitives.

Our construction of a subvertible OWP substantiates the folklore knowledge that sufficient random padding can render cryptosystems vulnerable to backdoor attacks, e.g., [27,28]. Specifically, the random padding in the malicious implementation can be generated so that it encrypts the corresponding trapdoor using the backdoor as a key. For detailed proofs, we defer to the full version [22].

Lemma 4. *One can construct a subvertible OWP from any TDOWP. In particular, given a TDOWP, we can construct a OWP that is not a OWP^{C}.*

We defer the question of the existence of a OWP$^{\text{C}}$ to the next section, where we will construct permutations that satisfy a stronger property.

Subversion-resistant OWPs with adversarially chosen indices: OWP$^{\mathbf{A}}$. The notion of OWP$^{\text{C}}$ formulated above defends against kleptographic attacks when the adversary provides a subverted implementation of the defining algorithms. In many cases, however, it is interesting to consider a more challenging setting where the adversary may directly provide the public parameters, including the function index. Indeed, this is the case in many real-world deployment settings, where a “trusted” agency sets up (or recommends) the public parameters. One notorious example (for a different primitive) is the Dual_EC PRG [7]. Note that, in general, this notion is not very suitable for asymmetric key primitives, e.g. TDOWP, since allowing the adversary to set up the public key gives him the chance to generate the trapdoor. We will focus on OWP$^{\text{A}}$.

Definition 4. *A one-way permutation family $\mathcal{F} = \{f_i : X_i \to X_i\}_{i \in I}$ with the specification $\mathcal{F}_{\text{SPEC}} = (\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}})$, is* ***subversion-resistant***$^{\text{A}}$ *in the offline watchdog model, if there is a* PPT *watchdog $\mathcal{W}$, such that: for any* PPT *adversary $\mathcal{A}$ playing the following game (Fig. 2) with the challenger $\mathcal{C}$, either the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is non-negligible, or the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible.*
Here the detection probability *of the watchdog $\mathcal{W}$ with respect to $\mathcal{A}$ is defined as:*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{Eval}_{\text{IMPL}}}(1^\lambda, i_\bullet) = 1] - \Pr[\mathcal{W}^{\mathsf{Eval}_{\text{SPEC}}}(1^\lambda, i) = 1]\right| ,$$

and the advantage *of the adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{\mathsf{Eval}_{\text{IMPL}}}(1^\lambda, i_\bullet)) = 1\right] ,$$

where $i \leftarrow \mathsf{KG}_{\text{SPEC}}(1^\lambda)$, and $i_\bullet$ is chosen by the adversary.
We also say that such $\mathcal{F}_{\text{SPEC}}$ is a OWP^A in the offline watchdog model.

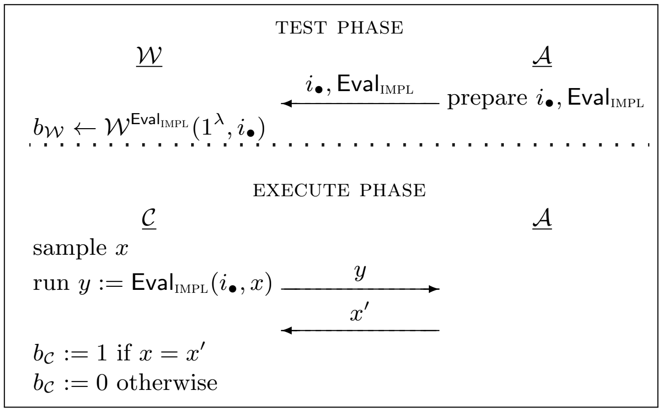

TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$i_\bullet, \mathsf{Eval}_{\text{IMPL}}$
prepare $i_\bullet, \mathsf{Eval}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{Eval}_{\text{IMPL}}}(1^\lambda, i_\bullet)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
sample x
run $y := \mathsf{Eval}_{\text{IMPL}}(i_\bullet, x)$
y
x'
$b_{\mathcal{C}} := 1$ if $x = x'$
$b_{\mathcal{C}} := 0$ otherwise

Fig. 2. Subversion-resistant$^{\text{A}}$ security game: OWP$^{\text{A}}$.

Relating OWP$^{\text{C}}$ and OWP$^{\text{A}}$. Following Lemma 3, an adversary that successfully breaks the OWP$^{\text{C}}$ game can be easily transformed into an adversary that breaks the OWP$^{\text{A}}$ game; thus *any OWP*$^{\text{A}}$ *is also a OWP*$^{\text{C}}$. As far as existence is concerned, then, it suffices to construct a OWP$^{\text{A}}$.

3.2 Constructing subversion-resistant$^{\text{A}}$ OWP

In this section, we discuss methods for safeguarding OWP against kleptographic attacks. We first present a general approach that transforms any OWP to a OWP$^{\text{A}}$ under the assumption that a suitable hash function can be defined on the index space. Specifically, we prove that randomizing the function index (via hashing, say) is sufficient to eliminate potential backdoor information. These results assume only the weakest (offline) watchdog. More importantly, we permit the hash function—like the other relevant cryptographic elements—to be implemented and potentially subverted by the adversary.

Note that we treat only the specification of the hash function in the random oracle model, assuming that the adversary may arbitrary subvert the (randomly specified) hash function; thus the watchdog is provided both the adversary's arbitrarily subverted "implementation" and the correct (random) hash function "specification."[9] Despite the adversary's control over the OWP and the hash function (which is partially constrained by the watchdog), it is difficult for the adversary to arrange a backdoor that works for a large enough target subset of function indices that these can be reliably "hit" by the hash function.

One remaining difficulty is to keep the "syntax intact," that is, to avoid changing the structure of the specification. For this purpose, we propose to treat the hash function only as a component of (jumping ahead) the evaluation algorithm (see Fig. 3). The adversary only implements the evaluation algorithm as a whole with the hash function built in (as the specification demands). In this case, the hash implementation (and specification) are not explicitly given to the watchdog anymore. However, we still manage to show the security by exploring the fact that both hash and the evaluation algorithm are deterministic algorithms with public input distribution, so that the offline watchdog can force the implementation of $\mathsf{Eval}_{\text{IMPL}}$ to agree with the specification $\mathsf{Eval}_{\text{SPEC}}$ with overwhelming probability when inputs are sampled according to the input generation distribution.

General feasibility results for OWP$^{\text{A}}$. Let $\mathcal{F}$ be any OWP family with specification $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}})$; while we assume, of course, that it is one-way secure (in the classical sense), it may be subvertible. We also assume that $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$ outputs uniform i from the index set I_λ and that we have a public hash function with specification $h_{\text{SPEC}} : I_\lambda \to I_\lambda$, acting on this set. Then we construct a subversion-resistant$^{\text{A}}$ OWP family $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}})$ defined as follows:

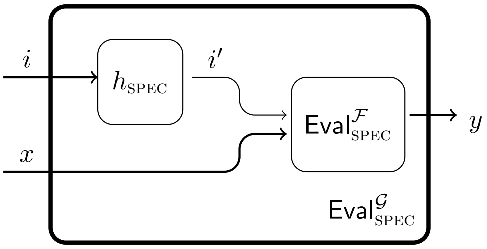

i
h_{SPEC}
i'
x
$\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}$
y
$\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}$

Fig. 3. New specification $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}$.

- Function index generation $i \leftarrow \mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$, where $\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$ is given by: Sample $i \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$; output i.

[9] Note that we place no a priori constraints on the subverted hash function provided by the adversary. The watchdog, of course, can ensure that the subverted function and the specification (which is just a random function, in this case) agree with high probability on slices of the space, or possess other common statistical properties.

- Function evaluation $y \leftarrow \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i,x)$, where $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}$ is given by: Upon receiving inputs (i,x), compute $i' = h_{\text{SPEC}}(i)$ and compute $y = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i',x)$; output y. See also the pictorial illustration for $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}$ in Fig. 3.

Remark 2. Note that the specification of the hash function is "part of" of the specification of the evaluation function. In fact, an interesting property of the construction above is that it is secure even if the (subverted) hash function is not separately provided to watchdog.[10]

Security analysis. Roughly, the proof relies on the following two arguments: (1.) any particular adversary can only invert a sparse subset of the one-way permutations; otherwise, such an adversary could successfully attack the (classical) security of the specification OWP. Thus, randomizing the function index will map it to a "safe" index, destroying the possible correlation with any particular backdoor. (2.) The $\mathsf{Eval}_{\text{IMPL}}$ (having the hash function h_{IMPL} built in) is a *deterministic* function that is only called on fixed public input distributions ($\mathcal{I} \times \mathcal{U}$, where $\mathcal{I}$ is the output distribution of $\mathsf{KG}_{\text{IMPL}}$ and $\mathcal{U}$ is the uniform distribution over the input space, and both of them are known to the watchdog). Following Lemma 1 of Section 2, the watchdog can ensure that $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ is consistent with its specification an overwhelming probability when inputs are generated according to $\mathcal{I} \times \mathcal{U}$. We remark that on all inputs for which the hash implementation (running inside $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$) is consistent with h_{SPEC}, random oracle queries have to be made.

Theorem 1. *Assume h_{SPEC} is random oracle, and $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}}$ is a OWP. Then $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}}$ defined above is a OWP$^{\text{A}}$ in the offline watchdog model.*

Proof. Suppose that $\mathcal{G}$ is not subversion-resistant$^{\text{A}}$. Then for any watchdog $\mathcal{W}$, there is a PPT adversary $\mathcal{A}_{\mathcal{G}}$ so that the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}_{\mathcal{G}}}$ is negligible and the advantage $\mathbf{Adv}_{\mathcal{A}_{\mathcal{G}}}$ is non-negligible, say δ. We will construct an adversary $\mathcal{A}_{\mathcal{F}}$ which will break the one-way security of $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}})$ with non-negligible probability. In particular, we define a simple watchdog algorithm that samples a uniform input x, and compares whether $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i_\bullet, x) = \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i_\bullet, x)$, where $i_\bullet$ is the public parameter chosen by the adversary $\mathcal{A}_{\mathcal{G}}$. (Note that the evaluation of $\mathsf{Eval}_{\text{SPEC}}$ may involve querying random oracle.)

Construction of $\mathcal{A}_{\mathcal{F}}$. Suppose (i^*, y^*) are the challenges that $\mathcal{A}_{\mathcal{F}}$ receives from the challenger $\mathcal{C}_{\mathcal{F}}$ (the challenger for one way security of $\mathcal{F}_{\text{SPEC}}$), where $y^* = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i^*, x^*)$ for a randomly selected x^*. $\mathcal{A}_{\mathcal{F}}$ simulates a copy of $\mathcal{A}_{\mathcal{G}}$. In addition $\mathcal{A}_{\mathcal{F}}$ simulates the subversion-resistant$^{\text{A}}$ OWP game with $\mathcal{A}_{\mathcal{G}}$.

Before receiving the function index $i_\bullet$ and the implementation $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ from $\mathcal{A}_{\mathcal{G}}$, the adversary $\mathcal{A}_{\mathcal{F}}$ (also acting as the challenger in the OWP$^{\text{A}}$ game playing with $\mathcal{A}_{\mathcal{G}}$) operates as follows: First, note that h_{SPEC} is random oracle; whenever $\mathcal{A}_{\mathcal{G}}$ wants to evaluate h_{SPEC} on some points (or implementing the component for

[10] In general, development of secure primitives in the complete subversion model would presumably be easier if the watchdog can separately "check" the implementation of h even though we do not need this for the above construction.

$\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ that is consistent with h_{SPEC} for those points), $\mathcal{A}_{\mathcal{G}}$ has to make random oracle queries. Without loss of generality, assume $\mathcal{A}_{\mathcal{G}}$ asks q random oracle queries $i_1, \ldots, i_q$ where $q = \text{poly}(\lambda)$. Here $\mathcal{A}_{\mathcal{F}}$ randomly chooses a bit b to decide whether to embed i^* in the answers of random oracle queries at this stage. If $b = 0$, $\mathcal{A}_{\mathcal{F}}$ randomly selects an index $t \in \{1, \ldots, q\}$, and sets i^* to the answer for $h_{\text{SPEC}}(i_t)$; for all others $j \in \{1, \ldots, q\} \setminus \{t\}$, $\mathcal{A}_{\mathcal{F}}$ uniformly samples i'_j from the index set I and sets $h_{\text{SPEC}}(i_j) = i'_j$. If $b = 1$, for all $j \in \{1, \ldots, q\}$, the adversary $\mathcal{A}_{\mathcal{F}}$ uniformly samples i'_j from the index set I and sets $h_{\text{SPEC}}(i_j) = i'_j$.

After receiving $i_\bullet, \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ from $\mathcal{A}_{\mathcal{G}}$, if $b = 1$ the adversary $\mathcal{A}_{\mathcal{F}}$ sets i^* to $h_{\text{SPEC}}(i_\bullet)$; otherwise, it chooses a random value and sets that to $h_{\text{SPEC}}(i_\bullet)$. Next, $\mathcal{A}_{\mathcal{F}}$ gives y^* to $\mathcal{A}_{\mathcal{G}}$ as the challenge and receives an answer x' from $\mathcal{A}_{\mathcal{G}}$. Note that in this phase, whenever $\mathcal{A}_{\mathcal{G}}$ makes random oracle queries on i, if $i \in \{i_1 \ldots, i_q\} \cup \{i_\bullet\}$, it is provided the previous response as answer; otherwise, i' is randomly chosen in the index set I and is returned as the answer.

Last, $\mathcal{A}_{\mathcal{F}}$ checks whether $b = 0 \wedge i_\bullet \neq i_t$, or $b = 1 \wedge i_\bullet \in \{i_1, \ldots, i_q\}$ (in those cases, $\mathcal{A}_{\mathcal{F}}$ fails to embed i^* into the right value); if yes, $\mathcal{A}_{\mathcal{F}}$ aborts; otherwise, $\mathcal{A}_{\mathcal{F}}$ submits x' to challenger $\mathcal{C}_{\mathcal{F}}$ as his answer.

Probabilistic analysis. Now we bound the success probability of $\mathcal{A}_{\mathcal{F}}$. Suppose x^* is the random input chosen by $\mathcal{C}_{\mathcal{F}}$; let W denote the event that $\mathcal{A}_{\mathcal{F}}$ aborts, W_1 the event that $b = 0 \wedge i \neq i_t$, and W_2 the event that $b = 1 \wedge i \in \{i_1, \ldots, i_q\}$. Recall that $\Pr[x' = x^*] = \Pr[x' = x^* \mid \overline{W}] \Pr[\overline{W}]$. Let $Q = \{i_1 \ldots, i_q\}$.

We first bound $\Pr[\overline{W}]$. Note that $\Pr[\overline{W}] = 1 - \Pr[W]$, and $\Pr[W] = \Pr[W_1 \vee W_2] \leq \Pr[W_1] + \Pr[W_2]$. Assuming $\Pr[i_\bullet \in Q] = \eta$, it follows that:

$$\begin{aligned} \Pr[W_1] &= \Pr[b = 0 \wedge i_\bullet \neq i_t] = \Pr[b = 0] \cdot \Pr[i_\bullet \neq i_t] \\ &= (1/2)(\Pr[i_\bullet \neq i_t \mid i_\bullet \in Q] \Pr[i_\bullet \in Q] + \Pr[i_\bullet \neq i_t \mid i_\bullet \notin Q] \Pr[i_\bullet \notin Q]) \\ &= (1/2)\left[(1 - (1/q)) \cdot \eta + (1 - \eta)\right] = (1/2)(1 - \eta/q)\,. \end{aligned}$$

While $\Pr[W_2] = \Pr[b = 1] \cdot \Pr[i \in Q] = \eta/2$, we have: $\Pr[W] \leq (1/2)(1 - (\eta/q) + \eta) = (1/2)(1 + \eta(1 - 1/q)) \leq (1/2)(1 + 1 \cdot (1 - 1/q)) = 1 - 1/(2q)$. Thus we can derive that $\Pr[\overline{W}] \geq 1/(2q)$.

Next, we bound $\Pr[x' = x^* | \overline{W}]$. From the assumption that $\mathcal{A}_{\mathcal{G}}$ breaks the security of $\mathcal{G}$, we have the following two conditions: (1) the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}_{\mathcal{G}}}$ is negligible; (2) the advantage $\mathbf{Adv}_{\mathcal{A}_{\mathcal{G}}}$ is non-negligible δ.

From condition (1), we claim: $\Pr[\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i_\bullet, x) = \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i_\bullet, x)] \geq 1 - \mathsf{negl}(\lambda)$. The probability is over the choices of x from uniform distribution over the input space. If not, the portion of inputs that $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ deviates from its specification is non-negligible (say, δ_1) in the whole domain. The watchdog $\mathcal{W}$ we defined (that simply samples an x and tests if the values $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i_\bullet, x)$ and $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i_\bullet, x)$ are equal) satisfies that $\Pr[\mathcal{W}^{\mathsf{Eval}_{\text{IMPL}}}(1^\lambda, i_\bullet) = 1] = 1 - \delta_1$. On the other hand, $\Pr[\mathcal{W}^{\mathsf{Eval}_{\text{SPEC}}}(1^\lambda, i) = 1] = 1$. This implies that $\mathbf{Det}_{\mathcal{W},\mathcal{A}_{\mathcal{G}}}$ is δ_1, which contradicts condition (1). Conditioned on $\overline{W}$, the equalities:

$$y^* = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i^*, x^*) = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(h_{\text{SPEC}}(i_\bullet), x^*) = \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i_\bullet, x^*) = \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i_\bullet, x^*)$$

hold with an overwhelming probability. That said, conditioned on $\overline{W}$, from $\mathcal{A}_{\mathcal{G}}$'s view, the distribution of y^* is identical to what she expects as a challenge in the subversion-resistant$^{\mathrm{A}}$ game.

Recall now from condition (2) the advantage $\mathbf{Adv}_{\mathcal{A}_{\mathcal{G}}}$ is non-negligible δ; this means $\mathcal{A}_{\mathcal{G}}$ inverts challenge $y^* = \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i_\bullet, x^*)$ and returns a correct $x' = x^*$ with probability δ. Combining the above, we can conclude that:

$$\Pr[x' = x^*] \geq \delta(1 - \mathsf{negl}(\lambda))\frac{1}{2q} = \frac{\delta}{2q} - \mathsf{negl}(\lambda)$$

which is non-negligible; note that $q = \mathrm{poly}(\lambda)$. Thus $\mathcal{A}_{\mathcal{F}}$ breaks the security of $\mathcal{F}_{\text{SPEC}}$, which leads to a contradiction. $\square$

3.3 Constructing subversion-resistant$^{\mathbf{SP}}$ OWP/TDOWP

We can define the notion of subversion-resistant$^{\mathrm{C}}$ TDOWP similar as OWP$^{\mathrm{C}}$. (Note that a *subvertible* TDOWP means that the adversary can invert the TDOWP using a backdoor which may have no relation to the regular trapdoor.) We defer the formal definition to the full version [22].

Indices (names) of a OWP family may have structure. For example, for OWPs based on discrete logarithm, $f_{g,p}(x) = g^x \bmod p$, the function index consists of an algebraically meaningful pair (p, g), where p is a prime and g a random generator. Applying the immunization strategy above would then require a hash function that respects this algebraic structure, mapping meaningful pairs (g, p) to meaningful pairs (g', p'). Furthermore, for a TDOWP, we must assume there is a public algorithm that can map between (public key, trapdoor) pairs.

To address these difficulties, we propose a practical *split-program* model in which every function generation algorithm (and, in general, any randomized algorithm) is composed of two components: a "random string generation algorithm" RG that outputs a uniform ℓ-bit string r, and a deterministic function index generation algorithm dKG that transforms the randomness r into a function index i. In this model, dKG is deterministic and is coupled with a known public input distribution (the output distribution of RG). Following Lemma 1 and the elaboration in Section 3.1, a watchdog can ensure that the implementation $\mathsf{dKG}_{\text{IMPL}}$ is "almost consistent" with $\mathsf{dKG}_{\text{SPEC}}$ (the specification) over this input distribution, i.e., $\Pr[\mathsf{dKG}_{\text{IMPL}}(r) = \mathsf{dKG}_{\text{SPEC}}(r) : r \leftarrow \mathsf{RG}_{\text{IMPL}}] \approx 1$. Morally, this forces the adversary to concentrate his malicious efforts on subverting RG.

Since we already demonstrated how to analyze the immunizing strategy for OWP, in this section we present results for TDOWP$^{\mathrm{SP}}$. It is straightforward to adapt the construction and analysis to OWP$^{\mathrm{SP}}$. The standard TDOWP definitions can be easily adapted in the split-program model, where the challenge index is generated by running $\mathsf{dKG}_{\text{SPEC}}$ on a uniform string r generated by $\mathsf{RG}_{\text{SPEC}}$. It is easy to see that a standard TDOWP is also a TDOWP in the split program model. For detailed definition, we refer to the full version [22].

Next we define the notion of a subversion-resistant$^{\mathrm{SP}}$ TDOWP in the split-program model by simply augmenting Definition 3. It is easy to see the same method applies to $\mathrm{OWP}^{\mathrm{SP}}$ as well. For detailed discussions of $\mathrm{OWP}^{\mathrm{SP}}$, we defer to the full version.

Generic construction of $\mathbf{TDOWP}^{\mathrm{SP}}$. Consider a TDOWP family $\mathcal{F}$ with specification $\mathcal{F}_{\mathrm{SPEC}} := (\mathsf{RG}^{\mathcal{F}}_{\mathrm{SPEC}}, \mathsf{dKG}^{\mathcal{F}}_{\mathrm{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\mathrm{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\mathrm{SPEC}})$, where $\mathsf{RG}^{\mathcal{F}}_{\mathrm{SPEC}}$ outputs uniform bits. Assuming a public hash function with specification $h_{\mathrm{SPEC}} : \{0,1\}^* \to \{0,1\}^*$, we construct a $\mathrm{TDOWP}^{\mathrm{SP}}$ family $\mathcal{G}$ with specification $\mathcal{G}_{\mathrm{SPEC}} = (\mathsf{RG}^{\mathcal{G}}_{\mathrm{SPEC}}, \mathsf{dKG}^{\mathcal{G}}_{\mathrm{SPEC}}, \mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}, \mathsf{Inv}^{\mathcal{G}}_{\mathrm{SPEC}})$, defined below:

- Randomness generation $r \leftarrow \mathsf{RG}^{\mathcal{G}}_{\mathrm{SPEC}}$: $\mathsf{RG}^{\mathcal{G}}_{\mathrm{SPEC}}$ is the same as $\mathsf{RG}^{\mathcal{F}}_{\mathrm{SPEC}}$. That is, $\mathsf{RG}^{\mathcal{G}}_{\mathrm{SPEC}}$ runs $\mathsf{RG}^{\mathcal{F}}_{\mathrm{SPEC}}$ to get r and outputs r.
- Index/trapdoor generation algorithm $(i, t_i) \leftarrow \mathsf{dKG}^{\mathcal{G}}_{\mathrm{SPEC}}(r)$: Upon receiving inputs r, it computes $\tilde{r} \leftarrow h_{\mathrm{SPEC}}(r)$, and outputs $(i, t_i) \leftarrow \mathsf{dKG}^{\mathcal{F}}_{\mathrm{SPEC}}(\tilde{r})$.
- $\mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}, \mathsf{Inv}^{\mathcal{G}}_{\mathrm{SPEC}}$ are the same as $\mathsf{Eval}^{\mathcal{F}}_{\mathrm{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\mathrm{SPEC}}$.[11]

See also the pictorial description for $\mathsf{dKG}^{\mathcal{G}}_{\mathrm{SPEC}}$ in Fig. 4:

Security analysis. The security of $\mathrm{OWP}^{\mathrm{SP}}/\mathrm{TDOWP}^{\mathrm{SP}}$ is more subtle than it looks. Randomizing the function index directly indeed destroys any backdoor structure; however, simply randomizing the random coins for generating the function index might lead the adversary to another index/backdoor pair. It will be critical in the split-program model that, with an offline watchdog, the output of $\mathsf{RG}_{\mathrm{IMPL}}$ is unpredictable even to the adversary who implements it.

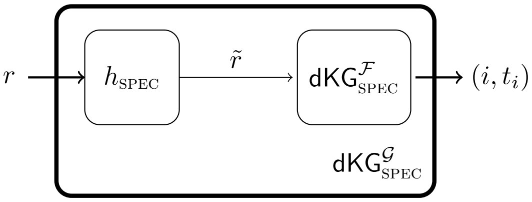

r
h_{SPEC}
$\tilde{r}$
$\mathsf{dKG}^{\mathcal{F}}_{\mathrm{SPEC}}$
(i, t_i)
$\mathsf{dKG}^{\mathcal{G}}_{\mathrm{SPEC}}$

Fig. 4. New specification $\mathsf{dKG}^{\mathcal{G}}_{\mathrm{SPEC}}$.

A few words about the security proof: Recall that in the $\mathrm{OWP}^{\mathrm{A}}$ proof, the reduction tries to "program the random oracle" so that the challenge of the specification can be embedded into the challenge to the adversary. In the split-program model, however, the reduction can directly embed the challenge if outputs of RG are unpredictable to the adversary; in this case, from the view of the adversary, any random index as challenge is possible to appear in the $\mathrm{TDOWP}^{\mathrm{SP}}$ game. Therefore, we here do not need to program the random oracle. We defer the full proof to the full version [22].

Theorem 2. *Assume h_{SPEC} is random oracle, and $\mathcal{F}$ with specification $\mathcal{F}_{\mathrm{SPEC}}$ is a TDOWP. Then $\mathcal{G}$ with specification $\mathcal{G}_{\mathrm{SPEC}}$ defined above is a $\mathrm{TDOWP}^{\mathrm{SP}}$ in the offline watchdog model.*

[11] We remark that in the split-program model, the hash function applies to the random bits, and the hash function is implemented by the adversary inside $\mathsf{Eval}^{\mathcal{G}}_{\mathrm{IMPL}}$. The specification of the hash function can be modeled as a random oracle so that replacing the random oracle with an explicit function like SHA256 may be heuristically justified.

4 Subversion-Resistant Signatures

In this section, we consider the challenge of designing digital signature schemes secure against kleptographic attacks. Previously results [1,4] suggest that a unique signature scheme [13,17] is secure against subversion of the signing algorithm assuming it satisfies the *verifiability condition*: every message signed by the sabotaged $\mathsf{Sign}_{\text{IMPL}}$ should be verified via $\mathsf{Verify}_{\text{SPEC}}$. As mentioned in the introduction, in these constructions *the key generation and verification algorithms are assumed to be faithfully implemented* while, in practice, all implementations normally come together. Thus, our goal in this section is to construct a signature scheme secure in the *complete subversion* model.

We emphasize that, in general, bringing a reduction between two primitives in the classical cryptographic world into the kleptographic world turns out to be highly non-trivial. We will see that the well-known reduction for full domain hash does not go through in the kliptographic setting when we try to build a subversion-resistant$^{\mathrm{C}}$ signature from a TDOWP$^{\mathrm{C}}$. (See Remark 3 and the proof of Theorem 3 for more details).

Following our general framework, it is easy to derive a definition for subversion-resistant signature scheme. As pointed out in [1], it is impossible to achieve unforgeability without the verifiability condition. Using our terminology, it is impossible to construct a subversion-resistant signature scheme with just an offline watchdog, even if only the Sign algorithm is subverted. So we will work in the online watchdog model where the watchdog can check the transcripts generated during the game between $\mathcal{C}$ and $\mathcal{A}$.[12] Next we define the security for digital signature schemes in the complete subversion model where all algorithms are implemented by the adversary, including the key generation algorithm.

Definition 5. *The specification $\Pi_{\text{SPEC}} = (\mathsf{KG}_{\text{SPEC}}, \mathsf{Sign}_{\text{SPEC}}, \mathsf{Verify}_{\text{SPEC}})$ of a signature scheme is **subversion-resistant**$^{\mathrm{C}}$ in the online watchdog model if there exists a* PPT *watchdog $\mathcal{W}$ such that: for any* PPT *adversary $\mathcal{A}$ playing the following game (Fig. 5) with the challenger $\mathcal{C}$, either the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is non-negligible, or the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible.*
Here the detection probability *of the watchdog $\mathcal{W}$ with respect to $\mathcal{A}$ is defined as*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\Pi_{\text{IMPL}}}(1^\lambda, \tau) = 1] - \Pr[\mathcal{W}^{\Pi_{\text{SPEC}}}(1^\lambda, \widehat{\tau}) = 1]\right| ,$$

and the advantage *of the adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{\Pi_{\text{IMPL}}}(1^\lambda)) = 1\right] ,$$

where τ is the transcript that generated when the challenger uses Π_{IMPL} and $\widehat{\tau}$ is the transcript generated when the challenger uses Π_{SPEC}.

[12] Note that, for digital signature schemes, it seems far preferable to adopt an online watchdog rather than an omniscient watchdog as in [4,9]. Due to the nature of signature schemes, transcripts consist of message-signature pairs which could arguably be *publicly* verified, and an online watchdog is sufficient.

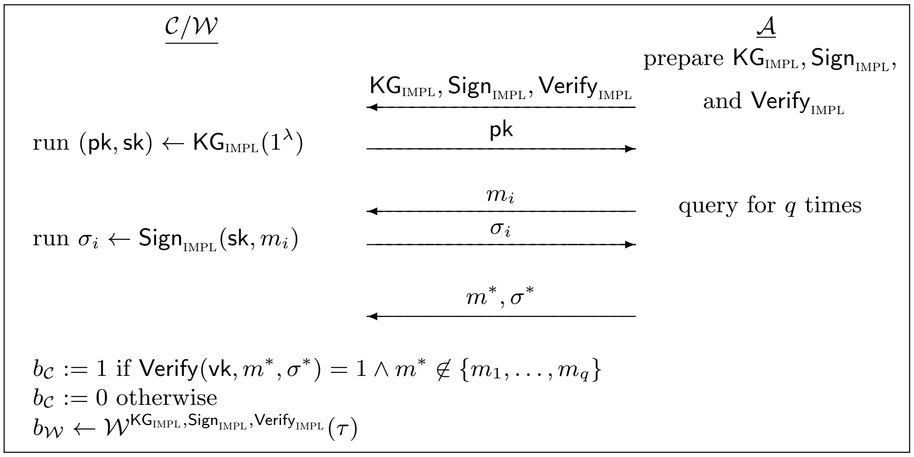

$\underline{\mathcal{C}/\mathcal{W}}$
$\underline{\mathcal{A}}$
prepare $\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}},$ and $\mathsf{Verify}_{\text{IMPL}}$
$\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}}, \mathsf{Verify}_{\text{IMPL}}$
run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}_{\text{IMPL}}(1^\lambda)$
pk
m_i
query for q times
run $\sigma_i \leftarrow \mathsf{Sign}_{\text{IMPL}}(\mathsf{sk}, m_i)$
σ_i
m^*, σ^*
$b_\mathcal{C} := 1$ if $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, \ldots, m_q\}$
$b_\mathcal{C} := 0$ otherwise
$b_\mathcal{W} \leftarrow \mathcal{W}^{\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}}, \mathsf{Verify}_{\text{IMPL}}}(\tau)$

Fig. 5. Subversion-resistant$^\text{C}$ Signature Game, where $\tau := (\mathsf{pk}, \{m_i, \sigma_i\}_{i \in [q]}, m^*, \sigma^*)$

Discussion. To extend previous results to the complete subversion model, the main challenge is to protect the (randomized) key generation algorithm against subversion attacks. While the subliminal channel attacks of Bellare et al. [4] apply to arbitrary sabotaged randomized algorithms, we observe that the key generation algorithm will be run only once (as in the security definition) which provides some hope that the subliminal channel can be controlled.

Next, we will prove that a variant of the widely deployed full domain hash scheme [8,5] can achieve the security in the complete subversion model. More concretely, in this variant, the signing algorithm needs to hash m *together with* pk; we remark that this modification is critical for the security reduction (see Remark 3). When instantiating its key generation with our subversion-resistant TDOWP$^\text{C}$, this variant gives a subversion-resistant signature scheme.

Constructing signature schemes with an online watchdog. Given a TDOWP$^\text{C}$, with specification $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}})$, and a public hash function with specification $h_{\text{SPEC}} : \mathcal{PK} \times \mathcal{M} \to \mathcal{M}$, where $\mathcal{PK}$ is the public key space and $\mathcal{M}$ is the message space, we construct a subversion-resistant$^\text{C}$ signature scheme $\mathcal{SS}$ with specification $\mathcal{SS}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}, \mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}, \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}})$ as follows:

- Key generation $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}(\lambda)$, where $\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}$ is given by: Compute $(i, t_i) \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$, and set $\mathsf{pk} := i$ and $\mathsf{sk} := t_i$;
- Signature generation $\sigma \leftarrow \mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}(\mathsf{pk}, \mathsf{sk}, m)$, where $\mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}} = (h_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}})$ is given by: Upon receiving message m, compute $\tilde{m} = h_{\text{SPEC}}(\mathsf{pk}, m)$, and $\sigma = \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}(\mathsf{pk}, \mathsf{sk}, \tilde{m})$, where $\mathsf{pk} = i, \mathsf{sk} = t_i$.
- Verification algorithm $b \leftarrow \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}(\mathsf{pk}, m, \sigma)$, where $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}} = (h_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}})$ is given by: Upon receiving message-signature pair (m, σ), if $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(\mathsf{pk}, \sigma) = h_{\text{SPEC}}(\mathsf{pk}, m)$ then set $b := 1$, otherwise set $b := 0$; here $\mathsf{pk} = i$.

Remark 3. We emphasize here that the specification of the Sign algorithm defines the hash and the inversion function separately, thus the adversary has to provide the implementation for each of them to the watchdog. Verify is treated similarly.

We also stress that using the full domain hash directly (without adding pk into the hash) results in the possibility that the random oracle query for m^* is asked before the implementations are prepared. In this case, the simulator has not yet received y^* from the TDOWP challenger, and the simulator has no way to embed y^* into the target. Including the pk in the hash essentially renders any random oracle queries made before the implementations are provided essentially useless (as they will be unrelated to any of the signatures), since the adversary cannot predict the actual value of pk. We defer the detailed proof to the full version [22].

Theorem 3. *Assume h_{SPEC} is random oracle, and $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}}$ is a TDOWP$^{\text{C}}$ in the offline watchdog model. Then the signature scheme $\mathcal{SS}$ with specification $\mathcal{SS}_{\text{SPEC}}$ constructed above is subversion-resistant$^{\text{C}}$ in the online watchdog model.*

5 Subversion-Resistant Pseudorandom Generators

Having studied the fundamental building blocks (OWPs and TDOWPs) in the complete subversion model, we now attempt to mimic the classical program of constructing richer cryptographic primitives from OWP/TDOWPs. We proceed in two different ways. The first is to carry "black-box" construction, the second is to generalize the immunizing strategy to broader settings. We remark that typical "black-box" constructions and reductions may not survive in the cliptographic model (indeed, even such basic features as the presence of multiple calls to a randomized algorithm can significantly affect security [4].) We begin by focusing on pseudorandom generators (PRG).

We first review the basic notions of PRG under subversion and then provide a specific construction that mimics the classical Blum-Micali PRG construction in this cliptographic context. We then examine how to extend the applicability of our general sanitizing strategy for OWP/TDOWPs to more general settings, demonstrating a strategy of public immunization for PRGs. Note that an impossibility result was shown in [10] that no public immunizing strategy is possible if it is applied to the output of the backdoored PRG, so a solution involves some trusted randomness is proposed. We also remark that all algorithms in our backdoor-free PRG construction—including the sanitizing function (which can be part of the KG algorithm in the specification)—can be subverted. Thus we provide the first PRG constructions secure in the complete subversion model.

We remark that since we follow the formalization of [10], the stretching algorithm is deterministic and stateful. In this case, the input distribution is evolving and not fixed, a universal watchdog cannot exhaust all those distributions. Fortunately, in the case of PRG stretching algorithm, we can still establish such a result, see security analysis of theorem 4.

5.1 Preliminaries: The definition of a subversion-resistant$^{\mathrm{A}}$ PRG

We adopt the definition from [10]: a pseudorandom generator consists of a pair of algorithms $(\mathsf{KG}, \mathsf{PRG})$, where KG outputs a public parameter pk and $\mathsf{PRG} : \{0,1\}^* \times \{0,1\}^\ell \to \{0,1\}^\ell \times \{0,1\}^{\ell'}$ takes the public parameter pk and an ℓ-bit random seed s as input; it returns a state $s_1 \in \{0,1\}^\ell$ and an output string $r_1 \in \{0,1\}^{\ell'}$. PRG may be iteratively executed; in the i-th iteration, it takes the state from the previous iteration s_{i-1} as the seed and generates the current state s_i and output r_i. We use $q - \mathsf{PRG}$ to denote the result of q iterations of PRG with outputs $r_1, \dots, r_q$ (each $r_i \in \{0,1\}^{\ell'}$).

They then define the notion of a *backdoored PRG* [10]: the adversary sets up a public parameter pk and may keep the corresponding backdoor sk. The output distribution $\mathsf{PRG}(pk, \mathcal{U})$ must still look pseudorandom to all algorithms that do not hold the backdoor sk (e.g., it fools the watchdog), where $\mathcal{U}$ is the uniform distribution; however, with sk, the adversary is able to distinguish the output from a uniform string, breaking the PRG.

The definition of a backdoored-PRG [10] is closely related to the subversion-resistant^A definition in our definitional framework, as the adversary is empowered to choose the "index" pk. Although there are several variants that all appear meaningful and interesting for PRG in the cliptographic settings, we will initially focus on the subversion-resistant$^{\mathrm{A}}$ PRG as the striking real world example of Dual_EC subversion is indeed in this model. Additionally, from Lemma 3, we remark that any PRG$^{\mathrm{A}}$ is a PRG$^{\mathrm{C}}$.

We first reformulate the definition of [10] in the subversion-resistant$^{\mathrm{A}}$ cliptographic framework: There exist "specification" versions of the algorithms and an offline watchdog. The parameter generation algorithm $\mathsf{KG}_{\text{SPEC}}$ has the requirement that the distribution of the adversarially generated public parameter must be indistinguishable from the output distribution of $\mathsf{KG}_{\text{SPEC}}$. Additionally, as the PRG algorithm is deterministic, and its input distribution is public, an offline watchdog can ensure that it is consistent with its specification $\mathsf{PRG}_{\text{SPEC}}$ on an overwhelming fraction of the inputs. The formal definitions are as follows:

Definition 6. *We say that a PRG (with the specification* $(\mathsf{KG}_{\text{SPEC}}, \mathsf{PRG}_{\text{SPEC}})$*) is* q**-subversion-resistant**$^{\mathrm{A}}$ *in the offline watchdog model if, there exists a* PPT *watchdog* $\mathcal{W}$ *such that: for any* PPT *adversary* $\mathcal{A}$ *playing the following game (Fig. 6) with the challenger* $\mathcal{C}$*, either the detection probability* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ *is non-negligible, or the advantage* $\mathbf{Adv}_{\mathcal{A}}$ *is negligible.*
Here the detection probability *of the watchdog* $\mathcal{W}$ *with respect to* $\mathcal{A}$ *is defined as*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{PRG}_{\text{IMPL}}}(1^\lambda, pk_\bullet) = 1] - \Pr[\mathcal{W}^{\mathsf{PRG}_{\text{SPEC}}}(1^\lambda, pk) = 1]\right| ,$$

and the advantage *of the adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \left|\Pr\left[(\mathcal{A}(1^\lambda) \Leftrightarrow \mathcal{C}^{\mathsf{PRG}_{\text{IMPL}}}(1^\lambda, pk_\bullet)) = 1\right] - \frac{1}{2}\right| .$$

where $pk \leftarrow \mathsf{KG}_{\text{SPEC}}(1^\lambda)$*, and* $\mathsf{PRG}_{\text{IMPL}}, pk_\bullet$ *are chosen by the adversary.*
We say that such PRG is a PRG$^{\mathrm{A}}$ *to stress that the public parameters are generated by the adversary.*

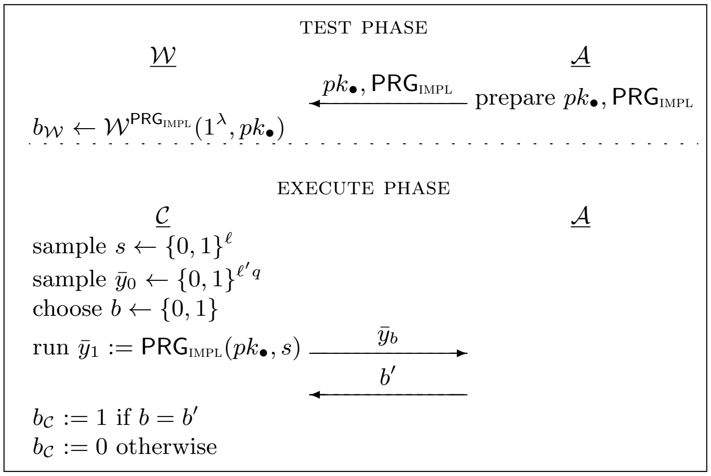

TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$pk_\bullet, \mathsf{PRG}_{\text{IMPL}}$
prepare $pk_\bullet, \mathsf{PRG}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{PRG}_{\text{IMPL}}}(1^\lambda, pk_\bullet)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
sample $s \leftarrow \{0,1\}^\ell$
sample $\bar{y}_0 \leftarrow \{0,1\}^{\ell' q}$
choose $b \leftarrow \{0,1\}$
run $\bar{y}_1 := \mathsf{PRG}_{\text{IMPL}}(pk_\bullet, s)$
$\bar{y}_b$
b'
$b_{\mathcal{C}} := 1$ if $b = b'$
$b_{\mathcal{C}} := 0$ otherwise

Fig. 6. Subversion-resistant$^{\text{A}}$ PRG Game

5.2 Constructing q-PRG^{A} from a OWP$^{\text{A}}$

In this section, we provide constructions of a PRG$^{\text{A}}$ based on a OWP$^{\text{A}}$. We start with a construction based on a (simplified) Blum-Micali PRG, and then extend it to a full-fledged solution. We remark that a similar reduction can be used to construct a subversion-resistant$^{\text{C}}$ PRG from a subversion-resistant$^{\text{C}}$ OWP (where the challenger queries $\mathsf{KG}_{\text{IMPL}}$ to choose a public parameter).

Before describing the details of our construction, we recall the classic generic construction of Goldreich-Levin (GL), yielding a hardcore predicate [12] for any OWF f. We suppose the input x of f is divided into two halves $x = (x_1, x_2)$ and define the bit $B(x) = \langle x_1, x_2 \rangle$; $B(x)$ is hard to predict given $x_1, f(x_2)$, assuming that f is one-way. Moreover, if there is a PPT algorithm that predicts $B(x)$ with significant advantage δ given $x_1, f(x_2)$, then there is a PPT algorithm I that inverts f with probability poly(δ).

Basic construction. We will show that given a subversion-resistant$^{\text{A}}$ one-way permutation (OWP) family $\mathcal{F}$ with specifications and implementations $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}})$ and $(\mathsf{KG}_{\mathcal{F}}, \mathsf{Eval}_{\mathcal{F}})$ respectively, the classic Blum-Micali PRG [6] (using the GL hardcore predicate) is 1-subversion-resistant$^{\text{A}}$. Our basic construction $\mathcal{G}$ with the specification $\mathcal{G}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}})$ is as follows:

- Parameter generation algorithm $pk \leftarrow \mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}(\lambda)$: compute $i \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$ and set $pk := i$;
- Bit string generation algorithm $(s', b) \leftarrow \mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}(pk, s)$: upon receiving s and pk, where $pk = i$, $s = s_1 || s_2$ and $|s_1| = |s_2| = \ell$, compute the following: $s'_1 := s_1$, $s'_2 := \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i, s_2)$, and $s' = s'_1 || s'_2$, $b := \langle s_1, s_2 \rangle$.

Security analysis. We can show in the lemma below that, with a specification designed as above, the basic construction above is a 1-subversion-resistant PRG. The intuition is that in the (simplified) Blum-Micali PRG, a distinguisher can be transformed into an OWP inverter (following the GL proof); thus an adversary who can build a backdoor for this PRG violates the subversion-resistance of $\mathcal{F}$. We present the lemma for its security, while due to lack of space, we refer the detailed proof to the full version [22].

Lemma 5. *If $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}}$ is a OWP$^{\text{A}}$ in the offline watchdog model, then $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}}$ constructed above is a 1-subversion-resistant$^{\text{A}}$ PRG in the offline watchdog model.*

Full-fledged PRG$^{\text{A}}$. We can easily adapt our basic construction to the full-fledged PRG$^{\text{A}}$ construction via the iteration as the BM-PRG and argue the security following the classic hybrid lemma. We refer the details of the construction and analysis to the full version [22].

5.3 A general public immunization strategy for PRG$^{\text{A}}$

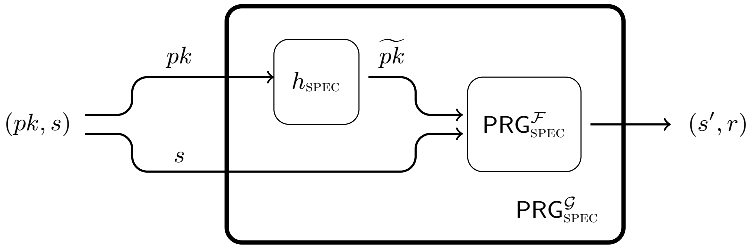
An impossibility result concerning public immunization of a PRG (to yield a PRG$^{\text{A}}$) was presented in [10]. However, we observe that this impossibility result only applies to an immunization procedure that operates on the *output* of the PRG$^{\text{A}}$. The general construction of OWP$^{\text{A}}$ shown above inspires us to consider an alternate general immunizing strategy for (potentially subvertible) PRGs. We establish that—similar to the procedure above for eliminating backdoors in OWPs—one can randomize the public parameter to sanitize a PRG. [13]

The intuition for this strategy to be effective in the setting of PRG is similar: considering a specification $\mathsf{KG}_{\text{SPEC}}$ that outputs a uniform pk from its domain, no single backdoor can be used to break the security for a large fraction of public parameter space; otherwise, one can use this trapdoor to break the PRG security of the specification. As above, while the adversary can subvert the hash function, an offline watchdog can ensure the hash function is faithful enough to render it difficult for the adversary arrange for the result of the hashed parameter to be amenable to any particular backdoor.

Consider a (potentially subvertible) PRG with specification $\mathcal{F}_{\text{SPEC}} = (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{PRG}^{\mathcal{F}}_{\text{SPEC}})$; we assume that $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$ outputs a uniform element of its range PP. Consider hash function with specification $h_{\text{SPEC}} : PP \to PP$. Then we construct a PRG$^{\text{A}}$ $\mathcal{G}$ with its specification $\mathcal{G}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}})$:

- Parameter generation algorithm $pk \leftarrow \mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$: Compute $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$, resulting in the output pk;
- Bit string stretch algorithm $(s', r) \leftarrow \mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}(pk, s)$ which is given by: Upon receiving a random seed s and public keys pk as inputs, it computes $\widetilde{pk} = h_{\text{SPEC}}(pk)$ and it computes $\mathsf{PRG}^{\mathcal{F}}_{\text{SPEC}}(\widetilde{pk}, s)$ and obtains s', r as outputs, where r would be the actual output, while s' would be used as the seed for next iteration. See also the pictorial illustration for $\mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}$ in Fig 7.

[13] To interpret this results, the solution of [10] is in a semi-private model which requires a trusted seed/key generation, thus part of the PRG algorithms can not be subverted. It follows that the construction of PRG in the complete subversion model was still open until our solution. In contrast, our sanitizing strategy does not require any secret, and even the deterministic hash function can be implemented by the adversary as part of the KG algorithm.



pk
h_{SPEC}
$\widetilde{pk}$
(pk, s)
s
$\mathsf{PRG}^{\mathcal{F}}_{\text{SPEC}}$
(s', r)
$\mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}$

Fig. 7. Public immunization strategy for PRG.

Security analysis. If the above PRG only iterates once, the security analysis would be very similar to that of Theorem 1; since any potential backdoor embedded in the public parameter is now destroyed, and the stretch algorithm is a deterministic algorithm with a public input distribution, thus an offline watchdog can already ensure it to be (essentially) consistent with its specification.

Things become trickier when the PRG may be iterated with arbitrary number of times. For example, suppose the watchdog checks only for t iterations, $\mathsf{PRG}_{\text{IMPL}}$ might deviate from the $t+1$-th iteration. This might be indeed problematic for general deterministic algorithms. Fortunately, for this particular example of PRG, the watchdog simply checks for one uniform input and compares the output with that generated by the specification is enough to ensure almost-everywhere consistency. To see this, the adversary can create a subset of inputs $S = \{s\}$, such that: $\mathsf{PRG}_{\text{IMPL}}(pk, s) \neq \mathsf{PRG}_{\text{SPEC}}(pk, s)$, where pk is the adversarially generated public parameter. Observe that the probability that a randomly chosen input s falls in S would be negligible. Otherwise the watchdog can detect with a non-negligible probability. While the difference with a stateful stretching algorithm is that it offers the adversary more chances to hit the bad set S because of the iterations. Note that when $\mathsf{PRG}_{\text{IMPL}}(pk, s) = \mathsf{PRG}_{\text{SPEC}}(pk, s)$ for some randomly chosen s, then the output s' would also be pseudorandom; iterating on this input, the stretching algorithm yields a polynomially many pseudorandom strings, thus the probability of any of those hit the bad set S would be still negligible. With this observation, we can still claim that with an overwhelming probability, $\mathsf{PRG}_{\text{IMPL}}$ will be consistent with $\mathsf{PRG}_{\text{SPEC}}$ even after arbitrary number of iterations (polynomially bounded). We defer the proof to the full version.

Theorem 4. *Assume h_{SPEC} is random oracle, and $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}} = (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{PRG}^{\mathcal{F}}_{\text{SPEC}})$ is a pseudorandom generator, where $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$ outputs pk randomly from its range. Then $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}}$ in the above construction yields a q-subversion-resistant PRG$^{\text{A}}$ for any polynomially large q.*

Remark 4. If the public parameter contains only random group elements, e.g., the Dual_EC PRG, we may simply encode them into bits and use a regular hash function like SHA-256, and convert the resulting bits back to a group element;

References

1. G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 364–375. ACM Press, Oct. 2015.
2. M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, Apr. 2015.
3. M. Bellare, J. Jaeger, and D. Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1431–1440. ACM Press, Oct. 2015.
4. M. Bellare, K. G. Paterson, and P. Rogaway. Security of symmetric encryption against mass surveillance. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, Aug. 2014.
5. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.
6. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, Nov. 1982.
7. S. Checkoway, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, H. Shacham, and M. Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 319–335, 2014.
8. J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, Aug. 2000.
9. J. P. Degabriele, P. Farshim, and B. Poettering. A more cautious approach to security against mass surveillance. In G. Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 579–598. Springer, Heidelberg, Mar. 2015.
10. Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, and T. Ristenpart. A formal treatment of backdoored pseudorandom generators. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, Apr. 2015.
11. Y. Dodis, I. Mironov, and N. Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. Cryptology ePrint Archive, Report 2015/548, 2015. `http://eprint.iacr.org/2015/548`.
12. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
13. S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, Aug. 1993.
14. I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 202–219. Springer, Heidelberg, Mar. 2009.
15. N. J. Hopper, J. Langford, and L. von Ahn. Provably secure steganography. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 77–92. Springer, Heidelberg, Aug. 2002.

16. A. Juels and J. Guajardo. RSA key generation with verifiable randomness. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 357–374. Springer, Heidelberg, Feb. 2002.
17. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.
18. I. Mironov and N. Stephens-Davidowitz. Cryptographic reverse firewalls. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, Apr. 2015.
19. NIST. Special publication 800-90: Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology, 2012. `http://csrc.nist.gov/publications/PubsSPs.html`.
20. N. Perlroth, J. Larson, and S. Shane. N.S.A. able to foil basic safeguards of privacy on web. The New York Times, 2013. `http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html`.
21. P. Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. `http://eprint.iacr.org/2015/1162`.
22. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Cliptography: Clipping the power of kleptographic attacks. Cryptology ePrint Archive, Report 2015/695, 2015. `http://eprint.iacr.org/2015/695`.
23. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Destroying steganography via amalgamation: Kleptographically cpa secure public key encryption. In *Cryptology ePrint Archive, Report 2016/530*, 2016. `http://eprint.iacr.org/2016/530`.
24. G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.
25. G. J. Simmons. A secure subliminal channel (?). In H. C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 33–41. Springer, Heidelberg, Aug. 1986.
26. Wikipedia. Nothing up my sleeve. `https://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number`.
27. A. Young and M. Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, Aug. 1996.
28. A. Young and M. Yung. Kleptography: Using cryptography against cryptography. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.
29. A. Young and M. Yung. Monkey: Black-Box symmetric ciphers designed for MONopolizing KEYs. In S. Vaudenay, editor, *FSE'98*, volume 1372 of *LNCS*, pages 122–133. Springer, Heidelberg, Mar. 1998.
30. A. Young and M. Yung. An elliptic curve backdoor algorithm for rsassa. In *Information Hiding*, IH '06, pages 355–374, 2006.
31. A. Young and M. Yung. Space-efficient kleptography without random oracles. In *Information Hiding*, IH '07, pages 112–129, 2007.
32. A. Young and M. Yung. Kleptography from standard assumptions and applications. In J. A. Garay and R. D. Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 271–290. Springer, Heidelberg, Sept. 2010.