Cliptography: Clipping the Power of Kleptographic Attacks

Alexander Russell[*] Qiang Tang[†] Moti Yung[‡] Hong-Sheng Zhou[§]

November 12, 2015

Abstract

Kleptography, introduced 20 years ago by Young and Yung [Crypto '96], studies how to steal information securely and subliminally from cryptosystems. The basic framework considers the (in)security of malicious implementations of a standard cryptographic primitives by embedding 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 attacks on symmetric key encryption algorithms, demonstrating a kleptographic attack that 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 *full subversion model*, where *all* relevant cryptographic primitives are subject to kleptographic attacks. We formally study 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. We augment this strategy with a "split program" model that can directly inform practical deployment.

We then examine two standard applications of (trapdoor) one-way permutations in this complete subversion model. First, we consider construction of "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. Second, we 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.

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.

[*]University of Connecticut, acr@cse.uconn.edu
[†]Cornell University, qt44@cornell.edu
[‡]Columbia University and Google, moti@cs.columbia.edu
[§]Virginia Commonwealth University, hszhou@vcu.edu

1 Introduction

Consider the conventional use of a cryptographic primitive, such as an encryption scheme: To encrypt a desired plaintext, one simply runs an implementation of the encryption algorithm obtained from a hardware or software provider with the plaintext as input. Although the underlying algorithms may be well-studied and proven secure, malicious implementations may cleverly 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. It is notable that *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 [YY96, YY97], the recent Snowden revelations [PLS13, LPS13] provided striking real-world examples that reawakened the security community to the seriousness of these issues. As a result, the topic has recently received renewed attention; see, e.g., [BPR14, BH15, DGG$^+$15, MS15, AMV15]. In particular, Bellare, Paterson, and Rogaway [BPR14] studied algorithm substitution attacks—with a focus on symmetric key encryption—and demonstrated a devastating framework for such attacks that apply in broad generality to randomized algorithms. Soon after, Dodis, Ganesh, Golovnev, Juels, and Ristenpart [DGG$^+$15] studied methods for "immunizing" pseudorandom generators (PRG) in such backdoored settings.

Our contributions. We continue this line of pursuit. 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 thus significantly amplifies previously studied settings, which rely on trusted key generation or clean randomness that is assumed private from the adversary. We study two fundamental cryptographic primitives in the complete subversion model—(trapdoor) one-way permutations (OWP)—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 *Related work* in Section 1. In detail, we show the following:

- We set down a hierarchy of three security models that capture practical kleptographic settings. The models are characterized by three parties: an adversary, who may provide potentially subverted implementations of cryptographic algorithms; a "watchdog," who either certifies or rejects the implementations by subjecting them to (black-box) interrogation [1]; and a challenger, who plays a conventional security game (using the potentially subverted algorithms) with the adversary. The role of the watchdog is thus the novel feature in the model: armed with the "specification" of the cryptographic algorithms and oracle access to the implementations provided by the adversary, he 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 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.

[1] Without the watchdog, it is elusive to achieve interesting cryptographic functionalities.

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* to make the general method above applicable using only standard hash functions. In the split-program model, the function generation algorithm is composed of two parts: a (randomized) randomness generation algorithm RG that outputs an (ostensibly) uniform bit string, and a (deterministic) function generation algorithm dKG that converts such random string into the function index. We remark that our results allow all algorithms to be implemented by the adversary; see Section 3.

- We then turn our attention to PRGs. Previous work of Dodis et al. [DGG+15] 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 applying 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 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. [DGG+15] 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 4.

- In Section 5, we then 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 digital signature scheme in the complete subversion model. More concretely, we achieve this result by (1) using the subversion-resistant trapdoor one way permutation 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 cliptographic setting. To resolve this issue, we slightly modify the FDH approach by hashing the message *together with the public key*, to make the proof go through. We remark that, in previous works, [BPR14, AMV15] demonstrated that a unique signature scheme is secure against kleptographic attacks, *assuming that the key generation algorithm is honest* and all the message-signature pairs can be checked by the lab/user. Our result is the first digital signature scheme allowing the adversary to sabotage all algorithms.

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: state and permitting use of randomized algorithms. We remark that our general defending technique differs from known methods: We here 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. 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 "stegochannel" attack of Bellare et al. [BPR14]. While our primitives do permit randomized algorithms, the security games we analyze invoke them only once (to, e.g., derive a key). The prospect of full "immunization" for general randomized algorithms is a—presumably challenging—direction of future work. This already improves the state-of-the-art of defending strategy of addressing only deterministic algorithms. In general, we focus on (potentially subverted) algorithms that are not permitted to maintain "state" between invocations; in some important cases noted in the paper, however, we can relax this assumption. (We mention that typical steganographic attacks, can indeed be carried out in a stateless model [BJK15].)

Related work. The concept of *kleptography*—subverting cryptographic algorithms by modifying their implementations to leak secrets covertly, was proposed by Young and Yung [YY96, YY97] in 1996. They gave concrete examples showing that backdoors can be embeded into the public keys of commonly used cryptographic schemes; while the resulting public keys appear normal to the users, the adversary is nevertheless capable of learning the secret keys. 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 [JG02], 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 "big brother" (adversary).

Bellare et al. considered a powerful family of kleptographic attacks that they call *algorithm substitution attacks*, and explore these in both symmetric key [BPR14] and public key [BH15] 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 covert subliminal channels [Sim83, Sim86, HLv02], and is particularly striking because it can be applied in such generality. 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 [DFP15]. 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. This forces us to manage certain randomized algorithms (such as key generation) in a kleptographic setting. The details of the model, with associated commentary about its relevance to practice, appear below.

Dodis et al. [DGG$^+$15] pioneered the rigorous study of pseudorandom generators in such settings, developing an alternative family of kleptographic attacks on pseudorandom generators in order to formalize the notorious Dual_EC PRG subversion [NIS12, CNE$^+$14]. 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 prove the equivalence of such a "backdoored PRG" and public key encryption with pseudorandom ciphertexts. Then they 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. We remark that our results on subversion-resistant OWFs can be applied to construct a specific "backdoor-free" PRG following the classic Blum-Micali framework. Moreover, our general immunizing strategy, randomizing the public parameter of a backdoored PRG instead of randomizing the PRG output, permits us to bypass an impossibility result established by Dodis et al. for general public immunization based on the PRG output.

Other works suggest different angles of defense against mass surveillance. For example, in [MS15, DMSD15], 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. [AMV15] 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 full 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 5.

2 A Definitional Framework for Cliptography

2.1 From Cryptography to Cliptography

In this section, we lay down a definitional framework for cliptography. As mentioned in the introduction, the new framework must reflect the ability of the adversary to provide (potentially subverted) implementations of the cryptographic primitives 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. In general, our 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.")

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 consider an even more powerful *omniscient* watchdog, which is even privy to private data of the challenger.

We remark that an offline watchdog cannot ensure that deterministic algorithms are faithfully implemented by the adversary; however, such a watchdog can ensure that they are correct with overwhelming probability for particular known distributions of inputs. Likewise, an offline watchdog can additionally ensure that distributions produced by the adversary's implementations are (computationally) indistinguishable from their "laboratory" counterparts. On the other hand, an online watchdog has significantly enhanced powers: he can now effectively ensure, for example, that all applications of a deterministic algorithm to arguments appearing in the transcript are correct. In many cases of interest, this is as good as ensuring that all deterministic functions have been faithfully implemented. The omniscient watchdog can actually provide such an iron-clad guarantee.

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 again the specification.

The complete subversion model. Another question 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 activities. (As remarked above, a strong watchdog can enforce significant constraints on the algorithms.) This contrasts with 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.

Stateless cryptographic schemes and classical security games. A cryptographic scheme Π consists of a set of (possibly randomized) algorithms $(F^1, \ldots, F^k)$ where each F^i is with input space $\mathcal{X}^i$, output space $\mathcal{Y}^i$, and randomness space $\mathcal{R}^i$. 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 a scheme $\Pi = (F^1, \ldots, F^k)$ results in a specification of the associated algorithms; for concreteness, we label these $(F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}})$; when a scheme is (perhaps adversarially) implemented, we denote the implementation as $\Pi_{\text{IMPL}} = (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$. If the implementation honestly follows the specification of the scheme, we denote the implementation as $\Pi_{\text{H-IMPL}} = (F^1_{\text{H-IMPL}}, \ldots, F^n_{\text{H-IMPL}})$. Note that, in principle, algorithms in the specification of a cryptographic scheme or implementations provided by an adversary could be stateful; in this paper, we focus on stateless algorithms/implementations.

Cryptographic games. We express the security requirements of cryptographic schemes via *cryptographic games* between a "challenger" and an "adversary."

Definition 2.1 (Cryptographic Game [HH09])**.** *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.*

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 truly random keys).

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 consider the other variants.

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 $b_{\mathcal{C}} = (\mathcal{A}(1^\lambda) \leftrightsquigarrow \mathcal{C}^{F^1_{\text{IMPL}},\ldots,F^k_{\text{IMPL}}}(1^\lambda))$ 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 Π provided by the adversary.)

As for the watchdog $\mathcal{W}$, the adversary provides $\mathcal{W}$ his potentially subverted implementations 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

$$b_{\mathcal{W}} = \mathcal{W}^{F^1_{\text{IMPL}},\ldots,F^k_{\text{IMPL}}}(1^\lambda)\,.$$

(Intuitively, the bit $b_{\mathcal{W}}$ indicates whether the implementations passed whatever tests the watchdog carried out to detect inconsistencies with the specification.)

Definition 2.2 (Cliptographic Game). *A cliptographic game $\widehat{\mathsf{G}} = (\mathcal{C}, \Pi_{\text{SPEC}}, \delta)$ is defined by a challenger $\mathcal{C}$, a primitive Π_{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 $\Pi_{\text{IMPL}} = (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$ denotes the implementation produced by $\mathcal{A}$. The advantage *of the adversary is defined to be*

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

*We say that a game is **subversion-resistant** if for all* PPT *adversaries $\mathcal{A}$, either there exists a watchdog $\mathcal{W}$ so that $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is non-negligible or $\mathbf{Adv}_{\mathcal{A}}$ is negligible.*

Remarks: random oracles. We remark that the definition of a cliptographic game requires setting down both the challenger *and* the specification of the primitive Π, as this latter data determines the notion of "detection probability." 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, then, 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 arbitrarily deviate from the random oracle itself. Likewise, during the security game, the challenger is provided oracle access only to the potentially subverted implementation of the random oracle. As usual, the probabilities defining the security (and detection) games are taken over the choice of the random oracle.

Discussion: the guarantees provided by an offline watchdog. We make some general observations about the guarantees that an offline watchdog can provide for deterministic algorithms.

First, consider a *deterministic* algorithm implemented by the adversary; an offline adversary cannot ensure that such an algorithm is perfectly implemented, but it can ensure that the implementation agrees with the specification with overwhelming probability with respect to any particular distributions of the watchdog's choice. In particular, in our security analysis we are free to assume that such testing has been carried out for any fixed family of distributions (perhaps involving sampling from other parts of the implementation).

Lemma 2.3. *Consider an adversarial implementation* $\Pi_{\text{IMPL}} := (F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$ *of a specification* $\Pi_{\text{SPEC}} = (F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}})$*, where* $F^1, \ldots, F^k$ *are deterministic algorithms. Additionally, for each* λ *define public input distributions* $X^1_\lambda, \ldots, X^k_\lambda$ *respectively. If for some* $j \in [k]$, $\Pr[F^j_{\text{IMPL}}(x) \neq F^j_{\text{SPEC}}(x) : x \leftarrow X^j_\lambda]$ *is non-negligible, there is an offline watchdog with detection probability* $1 - o(1)$*.*

Proof. Suppose there exists an implementation F^s_{IMPL} and a public input distribution X^s_λ so that $\Pr[F^s_{\text{IMPL}}(x) \neq F^s_{\text{SPEC}}(x) : x \leftarrow X^s_\lambda] \geq \delta$. Then a watchdog that simply checks equality on $\lceil \delta^{-1} \rceil$ samples will discover an inconsistency with constant probability. This can be amplified by repetition to achieve detection probability $1 - \mathsf{negl}(\lambda)$. We remark that the lemma does not require the implemented algorithms to be deterministic. $\square$

In our analysis, we will use this simple observation extensively. In particular, when a hash specification is modeled as a random oracle, on most of the input points, the implementation has to be consistent with it, and thus the corresponding random oracle queries have to be asked.

2.2.1 The online watchdog; the omniscient watchdog

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 such deterministic functions at only a few locations. 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. Observe that such inconsistencies are (essentially) undetectable to the watchdog; however, the adversary can ensure that the subverted algorithm is indeed queried at w during the security game. We remark that the above attack was noticed in various settings, e.g., input-triggering attack in [BPR14, DFP15, AMV15] that motivate them to consider the decryptability condition and verifiability condition.

An *online watchdog* $\mathcal{W}^{\text{online}}$ can guard against this possibility; an online watchdog is permitted to monitor the public interactions between users. More precisely, the online watchdog is permitted to certify both the implementations themselves *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 for the fact that the watchdog is provided the transcript τ [2] of the security game ($\mathcal{C} \leftrightsquigarrow \mathcal{A}$). (We use the shorthand notation Π_{IMPL} here to denote the full collection of oracles $F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}}$.) The detection game must then be adjusted, guaranteeing except negligible error, the transcripts that produced when the challenger uses Π_{IMPL} are indistinguishable from that when the challenger uses Π_{SPEC}. Our results on digital signature schemes will require such $\mathcal{W}^{\text{online}}$.

An *omniscient watchdog* $\mathcal{W}^{\text{omni}}$ 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 2.2 above with $\mathcal{W}^{\text{omni}}$, we obtain cliptographic games with omniscient watchdog. As mentioned, omniscient watchdog has been considered in literature [BPR14, DFP15]. While we do not require omniscient watchdog in our constructions in this paper, we discuss omniscient watchdog here as part of definitional framework.

2.2.2 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 2.2, 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 2.2 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.*

It is clear that if a primitive is secure in the adversarially chosen parameter model, then it is secure according to Definition 2.2 (even the implementation Gen is stateful). (Observe that the adversary is always free to generate pp according to the algorithm provided to the challenger.) We record this below.

Lemma 2.4. *If Π is secure in cliptographic definition in the adversarially chosen parameter model, then Π is secure according to Definition 2.2.*

2.2.3 The split-program model

Randomized algorithms play a distinguished role in our kleptographic setting. One technique we propose for immunization will 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 2.3, we may assume that the maliciously implemented $\mathsf{dKG}_{\text{IMPL}}$ is consistent with the honest implementation $\mathsf{dKG}_{\text{SPEC}}$ with overwhelming probability.

We remark that this perspective only requires a change in the specification of Π. When we apply Definition 2.2 with a specification that has been altered to reflect this split-program convention, we say that a primitive is proven secure in the *split-program model.*

Discussions. Intuitively, exposing the random coins offers the watchdog more flexibility. It seems a solution to the original scheme Π will immediately give us a solution to the modified scheme Π^{SP}.

[2] We remark that the transcript τ includes the final output bit of the challenger in the security game.

However, this is not necessary to be true. The adversary may also take advantage of the "exposed internal state" to break the scheme. See Section 3.3 for a concrete example.

The split-program model is quite general and can be applied to most practical algorithms. To see this, the user gets 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 interface with the calls to the API with the separate program RG provided by the big brother. In principle, one can always augment a randomized KG to output the function index i together with the coin r in this way.

3 Subversion-Resistant One-Way Permutations

In this section, we study one-way permutations (OWP) in our cliptographic framework. In particular, we propose general constructions for subversion-resistant OWPs that require only the weakest (offline) watchdog. Our "immunizing strategy" consists of coupling the function generation algorithm with a cryptographic 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. [3] 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. [4]

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 propose a natural "split-program" model that provides such uniformization 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.

Finally, the complete subversion 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 correspond 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 $\mathrm{OWP}^{\mathrm{SP}}$.

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.

[3]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.

[4]While for TDOWP, to maintain correctness, the hash is applied to the trapdoor, assuming there is a public deterministic procedure that generates the function index from a trapdoor.

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 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 3.1. *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, for any* PPT *adversary $\mathcal{A}$ playing with the challenger $\mathcal{C}$ in the following game, (Fig. 1), there exists a watchdog $\mathcal{W}$, such that: 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*

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

For convenience, we also say that such $\mathcal{F}_{\text{SPEC}}$ is a OWP^C in the offline watchdog model.

In addition, when this definition fails for a OWP, we say that it is subvertible.[5]

Subversion-resistant trapdoor OWPs. We define the notion of subversion-resistant$^{\text{C}}$ TDOWP analogously. (Note that a *subvertible* TDOWP means that the adversary can invert the TDOWP using a private backdoor which may have no relation to the regular trapdoor created by the challenger.) We defer the formal definition to Appendix A.

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

[5]To be notationally consistent, this could be called subvertible$^{\text{C}}$; however, the only example of a subvertible function we construct will be in this C model and, moreover, subvertibilty in this model will imply subvertibility in the other models.

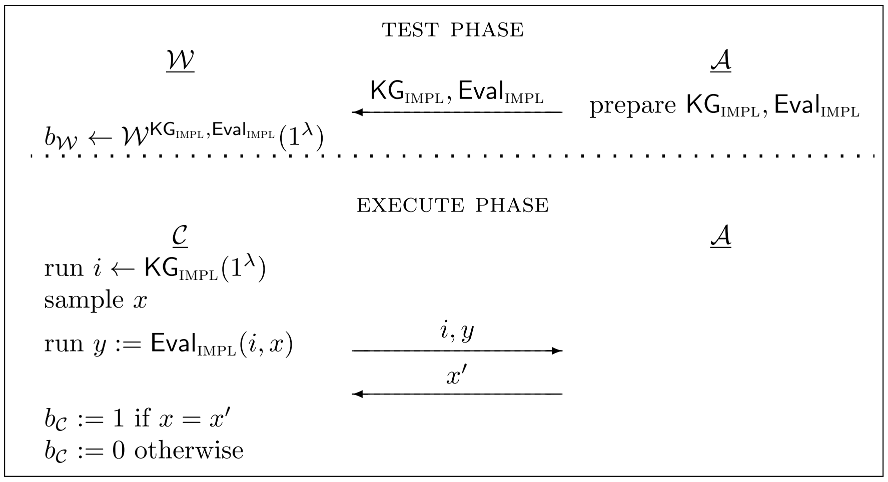
TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}$
prepare $\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}}(1^\lambda)$
EXECUTE 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

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

subvertible OWP (so the subverted implementation can evade detection by the watchdog 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., [YY96, YY97]. Specifically, the random padding in the malicious implementation can be generated so that it simply encrypts the corresponding trapdoor using the backdoor as a key. For detailed proofs, we defer to Appendix C.

Lemma 3.2. *One can construct a subvertible OWP from any TDOWP. In particular, given a TDOWP, we can construct a OWP that is not a OWP$^{\text{C}}$.*

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$^{\text{A}}$. 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 in a different setting is the Dual_EC PRG [CNE$^+$14]. Note that this notion is not very suitable for asymmetric key primitives, e.g. TDOWP, since allowing the adversary to set up the public key gives him the chance to generate the trapdoor. We will focus on OWP$^{\text{A}}$.

Definition 3.3. *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***$^{\text{A}}$ *in the offline watchdog model, if for any* PPT *adversary $\mathcal{A}$ playing the following game (Fig. 2) with the challenger $\mathcal{C}$, there exists a watchdog $\mathcal{W}$, such that: 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*

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

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

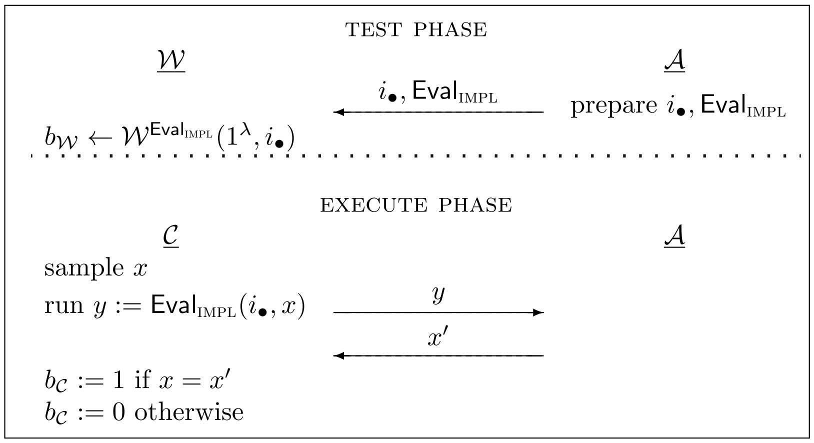
TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$i_\bullet, \mathsf{Eval}_{\text{IMPL}}$
prepare $i_\bullet, \mathsf{Eval}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{Eval}_{\text{IMPL}}}(1^\lambda, i_\bullet)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
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

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

Note that security games like this do not make much sense in the classical cryptographic setting; without the watchdog, the adversary can implement arbitrary functionalities so that the security can be trivially broken. As mentioned above, such security notions for OWP (or related symmetric key cryptographic primitives, like PRG [DGG$^+$15]) are meaningful in the cliptographic setting and model relevant practical settings.

Relating OWP$^{\text{C}}$ and OWP$^{\text{A}}$. Following Lemma 2.4, an adversary that successfully breaks the OWP$^{\text{C}}$ game can be easily transformed into an adversary that breaks the OWP$^{\text{A}}$ game; We can claim that *any OWP$^{\text{A}}$ is also a OWP$^{\text{C}}$*. As far as existence is concerned, then, it suffices to construct a OWP$^{\text{A}}$ (which satisfies the stronger subversion-resistant$^{\text{A}}$ condition).

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. Furthermore, 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."[6] 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 more difficulty left is that, since we are constructing OWP$^{\mathrm{A}}$, to keep the syntax intact, we propose to treat the hash function only as a component of (jumping ahead) the evaluation algorithm (see Fig. 3). The big brother only implements the evaluation algorithm as a whole with (or without) the hash function built in. 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 $\mathsf{Eval}_{\mathrm{IMPL}}$ to agree with the specification $\mathsf{Eval}_{\mathrm{SPEC}}$ with overwhelming probability when inputs are sampled according to the input generation distribution.

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

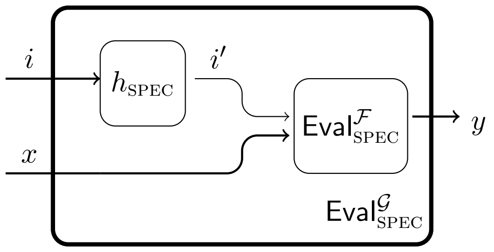
i
h_{SPEC}
i'
x
$\mathsf{Eval}^{\mathcal{F}}_{\mathrm{SPEC}}$
y
$\mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}$

Figure 3: New specification $\mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}$.

- Function index generation $i \leftarrow \mathsf{KG}^{\mathcal{G}}_{\mathrm{SPEC}}$, where $\mathsf{KG}^{\mathcal{G}}_{\mathrm{SPEC}}$ is given by:
  Sample $i \leftarrow \mathsf{KG}^{\mathcal{F}}_{\mathrm{SPEC}}(\lambda)$; output i.
- Function evaluation $y \leftarrow \mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}(i, x)$, where $\mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}$ is given by:
  Upon receiving inputs (i, x), compute $i' = h_{\mathrm{SPEC}}(i)$ and compute $y = \mathsf{Eval}^{\mathcal{F}}_{\mathrm{SPEC}}(i', x)$, output y.

See also the pictorial illustration for $\mathsf{Eval}^{\mathcal{G}}_{\mathrm{SPEC}}$ in Fig. 3.

Remark 3.4. *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.*[7]

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 the function index to a "safe" index, and destroy the possible correlation with any particular backdoor. (2.) The hash function and Eval are both *deterministic* functions that

[6] 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.

[7] 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.

are only called on fixed public input distributions (known to the watchdog). Following Lemma 2.3 of Section 2, the watchdog can ensure that $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ is consistent with its specification an overwhelming fraction of the inputs. 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 3.5. *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 there is a PPT adversary $\mathcal{A}_{\mathcal{G}}$ so that the detection probability **Det** is negligible and the advantage **Adv** 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.

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 $\mathcal{A}_{\mathcal{F}}$ 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 $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ 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 number of random oracle queries on $i_1, \ldots, i_q$ where $q = \text{poly}(\lambda)$. Here $\mathcal{A}_{\mathcal{F}}$ randomly chooses a bit b to decide whether to embed i^* to the answers of random oracle queries. If $b = 0$, $\mathcal{A}_{\mathcal{F}}$ randomly selects an index $t \in \{1, \ldots, q\}$, and sets i^* as 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^* as $h_{\text{SPEC}}(i_\bullet)$. 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 $i \in \{i_1 \ldots, i_q\} \cup \{i_\bullet\}$, then returns the previous response as answer; otherwise, randomly choose i' in the index set I, and return i' as the answer.

Last, $\mathcal{A}_{\mathcal{F}}$ checks whether $b = 0 \wedge i_\bullet \neq i_t$, or $b = 1 \wedge i_\bullet \in \{i_1, \ldots, i_q\}$; 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 $\Pr[x' = x^*] = \Pr[x' = x^* | \overline{W}] \Pr[\overline{W}]$. 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 $\Pr[i \in Q] = \eta$, it follows that:

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

While $\Pr[W_2] = \Pr[b=1]\Pr[i \in Q] = \eta/2$, we have:

$$\Pr[W] \leq (1/2)(1 - \eta/q + \eta) \leq (1/2)(1 - 1/q + 1) = 1 - 1/(2q)$$

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 **Det** is negligible; (2) the advantage **Adv** 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. 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. Then there always exists a watchdog $\mathcal{W}$ (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) so 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 **Det** is δ_1, which contradicts to 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 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$^{\text{A}}$ game, even if $\mathcal{A}_{\mathcal{G}}$ never queried random oracle on $i_\bullet$. Recall now from condition (2) the advantage **Adv** 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 $q = \text{poly}(\lambda)$. Thus $\mathcal{A}_{\mathcal{F}}$ breaks the security of $\mathcal{F}_{\text{SPEC}}$, which leads to a contradiction. □

One way functions and stateful implementations. To define a subversion-resistant$^{\text{A}}$ one way function, we can easily adjust definition 3.3 that the output bit of the challenger is decided by checking whether $\mathsf{Eval}_{\text{IMPL}}(i, x') = y$. Our construction above already works for immunizing one way function, if we are working in the online watchdog model. Observe that the current proof can be (almost) directly carried over, if the output bit is determined by the specification. Since evaluation is deterministic, an online watchdog can guarantee that x' is indeed a valid inversion.

Also, it is not hard to see that the above construction works even if the subverted implementations are stateful.

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

Indices (names) of a one-way function family may have structure. For example, for OWFs based on the 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. As mentioned previously, 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 compute the public key based on a trapdoor.

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 (sub-)algorithms: a "random string generation algorithm" RG that outputs a uniform ℓ-bit random 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 2.3 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. [8]

There are some subtle differences between the security of cliptographic constructions in the split-program model and those in the previous sections. Although exposing the random coins offers the watchdog improved access to the "internals" of the algorithm, it also gives the adversary extra opportunities. Randomizing the function index directly destroys the backdoor structure; however, simply hashing the randomness publicly does not provide defense against kleptographic attacks, in that the adversary simply uses the new randomness to generate a new backdoor. Thus it is critical in the split program model that the RG algorithm is *stateless* so that the watchdog can enforces its output to be unpredictable to the adversary (when $\mathsf{RG}_{\text{SPEC}}$ outputs uniform bits).

Since we already demonstrated how to analyze the immunizing strategy for OWP, in this section we present results for TDOWP^{SP}. It is straightforward to adapt the construction and analysis to OWP^{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 an 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 Definition A.2 in Appendix A.

Next we define the notion of a subversion-resistant$^{\text{SP}}$ TDOWP in the split-program model by simply augmenting Definition 3.1. It is easy to see the same method applies to OWP^{SP} as well. For detailed discussions of OWP^{SP} in the split program model, we defer to the full version.

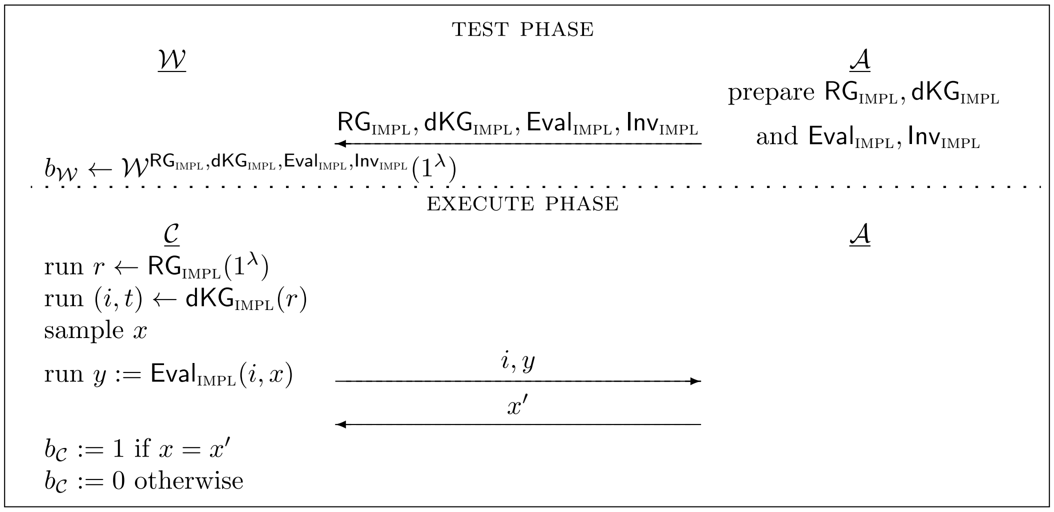
TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
prepare $\mathsf{RG}_{\text{IMPL}}, \mathsf{dKG}_{\text{IMPL}}$ and $\mathsf{Eval}_{\text{IMPL}}, \mathsf{Inv}_{\text{IMPL}}$
$\mathsf{RG}_{\text{IMPL}}, \mathsf{dKG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}, \mathsf{Inv}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{RG}_{\text{IMPL}}, \mathsf{dKG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}, \mathsf{Inv}_{\text{IMPL}}}(1^\lambda)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
run $r \leftarrow \mathsf{RG}_{\text{IMPL}}(1^\lambda)$
run $(i, t) \leftarrow \mathsf{dKG}_{\text{IMPL}}(r)$
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

Figure 4: Subversion-resistant$^{\text{SP}}$ TDOWP Game

[8]This gives us more flexibility to apply the sanitizing strategy. Incidentally, we can also decouple the immunizing strategy in the previous section (e.g., Fig 3) so that $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ explicitly implements h and $\mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$.

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

- Randomness generation $r \leftarrow \mathsf{RG}^{\mathcal{G}}_{\text{SPEC}}$:

  $\mathsf{RG}^{\mathcal{G}}_{\text{SPEC}}$ is the same as $\mathsf{RG}^{\mathcal{F}}_{\text{SPEC}}$. That is, $\mathsf{RG}^{\mathcal{G}}_{\text{SPEC}}$ runs $\mathsf{RG}^{\mathcal{F}}_{\text{SPEC}}$ to get r and outputs r.

- Index/trapdoor generation algorithm $(i, t_i) \leftarrow \mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}(r)$, which is given by:

  Upon receiving inputs r, it computes $\tilde{r} \leftarrow h_{\text{SPEC}}(r)$, and outputs $(i, t_i) \leftarrow \mathsf{dKG}^{\mathcal{F}}_{\text{SPEC}}(\tilde{r})$.

- $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{G}}_{\text{SPEC}}$ are the same as $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}$.[9]

See also the pictorial description for $\mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}$ in Fig. 5:

Security analysis. The security of OWP$^{\mathrm{SP}}$/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 that in the split-program model, the implementations provided by the adversary are stateless: With an offline watchdog, the output of a stateless RG is unpredictable even to the adversary who implements it.

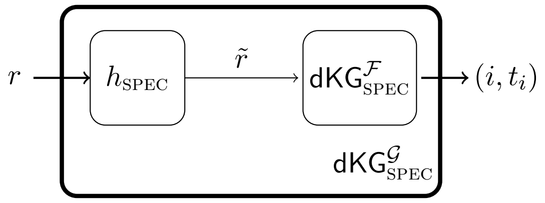
r
h_{SPEC}
$\tilde{r}$
$\mathsf{dKG}^{\mathcal{F}}_{\text{SPEC}}$
(i, t_i)
$\mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}$

Figure 5: New specification $\mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}$.

A few words about the security proof: Recall that in the 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 TDOWP$^{\mathrm{SP}}$ game. Therefore, we here do not need to program the random oracle.

Theorem 3.6. *Assume h_{SPEC} is random oracle, and $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}}$ is a TDOWP. Then $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}}$ defined above is a TDOWP$^{\mathrm{SP}}$ in the offline watchdog model.*

Proof. Suppose the is a PPT adversary $\mathcal{A}_{\mathcal{G}}$ that subvertes $\mathcal{G}$ in the game defined in Fig. 4. We will build an adversary $\mathcal{A}_{\mathcal{F}}$ that breaks the one-way security of $\mathcal{F}_{\text{SPEC}}$.

Construction of $\mathcal{A}_{\mathcal{F}}$. Assume (i^*, y^*) are the challenges sent from the $\mathcal{F}_{\text{SPEC}}$ challenger $\mathcal{C}_{\mathcal{F}}$ (suppose x^* is the input chosen by $\mathcal{C}_{\mathcal{F}}$ that generates this challenge). $\mathcal{A}_{\mathcal{F}}$ will simulate the game with $\mathcal{A}_{\mathcal{G}}$. When $\mathcal{A}_{\mathcal{G}}$ asks random oracle queries $r_1, \ldots, r_q$, $\mathcal{A}_{\mathcal{F}}$ answers all those queries with uniform strings.

$\mathcal{A}_{\mathcal{G}}$ then provides the implementations $(\mathsf{RG}^{\mathcal{G}}_{\text{IMPL}}, \mathsf{dKG}^{\mathcal{G}}_{\text{IMPL}}, \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}, \mathsf{Inv}^{\mathcal{G}}_{\text{IMPL}})$. $\mathcal{A}_{\mathcal{F}}$ continues the simulation, querying $\mathsf{RG}^{\mathcal{G}}_{\text{IMPL}}$ to receive r. If $r \in \{r_1, \ldots, r_q\}$, $\mathcal{A}_{\mathcal{F}}$ aborts; otherwise $\mathcal{A}_{\mathcal{F}}$ sends (i^*, y^*) to $\mathcal{A}_{\mathcal{G}}$. Finally, $\mathcal{A}_{\mathcal{F}}$ submits the answer x' from $\mathcal{A}_{\mathcal{G}}$ as his answer to $\mathcal{A}_{\mathcal{F}}$.

[9] 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 $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$. 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.

Probabilistic analysis. Now we bound the success probability of $\mathcal{A}_{\mathcal{F}}$. Suppose x is the random input chosen by $\mathcal{C}_{\mathcal{F}}$, and let W denote the event that $\mathcal{A}_{\mathcal{F}}$ aborts. It follows that: $\Pr[x^* = x'] = \Pr[x^* = x' | \overline{W}] \Pr[\overline{W}]$.

From the assumption that $\mathcal{A}_{\mathcal{G}}$ breaks the security of $\mathcal{G}$, we can infer that the following two conditions: (1) the detection probability **Det** is negligible; (2) the advantage **Adv** is non-negligible δ. From condition (1), we have

$$\Pr[r \notin \{r_1, \ldots, r_q\} : r \leftarrow \mathsf{RG}^{\mathcal{G}}_{\text{IMPL}}] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Otherwise, there is a watchdog algorithm that simply a sample to detect whether there is collision, in which case the implementation $\mathsf{RG}^{\mathcal{G}}_{\text{IMPL}}$ is rejected. On the other hand, if $\mathsf{RG}^{\mathcal{G}}_{\text{SPEC}}$ outputs uniform bits, the collision probability is negligible; note that here $\mathsf{RG}^{\mathcal{G}}_{\text{SPEC}} = \mathsf{RG}^{\mathcal{F}}_{\text{SPEC}}$. Thus $\Pr[\overline{W}] \geq 1 - \mathsf{negl}(\lambda)$.

Next, we bound $\Pr[x' = x^* | \overline{W}]$. From condition (1) again, following the proof of Lemma 2.3, we claim:

$$\Pr[\mathsf{dKG}^{\mathcal{G}}_{\text{IMPL}}(r) = \mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}(r) : r \leftarrow \mathsf{RG}^{\mathcal{F}}_{\text{IMPL}}] \geq 1 - \mathsf{negl}(\lambda); \tag{1}$$

$$\Pr[\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i^*, x) = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i^*, x)] \geq 1 - \mathsf{negl}(\lambda); \tag{2}$$

otherwise there is a trivial watchdog that samples, tests equality, and rejects misbehaving implementations.

Now let us analyze the view of the adversary $\mathcal{A}_{\mathcal{G}}$ in the subversion-resistant$^{\text{SP}}$ TDOWP game. From Ineq. (1), with overwhelming probability $\mathcal{A}_{\mathcal{G}}$ is supposed to see an index i in $(i, t) \leftarrow \mathsf{dKG}^{\mathcal{G}}_{\text{IMPL}}(r) = \mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}(r)$, where $r \leftarrow \mathsf{RG}^{\mathcal{G}}_{\text{IMPL}}$; similarly, $y \leftarrow \mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i, x)$ for a randomly selected x. While i^* is generated by calling $\mathsf{dKG}^{\mathcal{F}}_{\text{SPEC}}(r^*)$ for an uniform r^*. Conditioned on $\overline{W}$, the distribution of $\mathsf{dKG}^{\mathcal{G}}_{\text{SPEC}}(r) = \mathsf{dKG}^{\mathcal{F}}_{\text{SPEC}}(h_{\text{SPEC}}(r))$ is identical to $\mathsf{dKG}^{\mathcal{F}}_{\text{SPEC}}(r^*)$. Also, from Ineq. (2), with overwhelming probability, $y^* = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i^*, x^*)$ would be consistent with $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}(i^*, x^*)$. Thus we can claim that the distribution of i^*, y^* is indistinguishable from what $\mathcal{A}_{\mathcal{G}}$ expects. Together with condition (2), $\mathcal{A}_{\mathcal{G}}$ will return a correct x^* with a noticeable probability δ. Combined all above, we can conclude that

$$\Pr[x = x^*] \geq \delta(1 - \mathsf{negl}(\lambda))(1 - \mathsf{negl}(\lambda)) \geq \delta - \mathsf{negl}(\lambda)\,.$$

Thus $\mathcal{A}_{\mathcal{F}}$ breaks the security of $\mathcal{F}_{\text{SPEC}}$, a contradiction. □

4 Subversion-Resistant Pseudorandom Generators

Having studied the classical fundamental building blocks (OWPs and TDOWPs) in the full subversion model, we now attempt to mimic the classical program of constructing richer cryptographic primitives from OWP/TDOWPs. We remark that typical "black-box" constructions and reductions may not survive in the full subversion model (indeed, even such basic features as the presence of multiple calls to a randomized algorithm can significantly affect security [BPR14].) We begin by focusing on pseudorandom generators (PRG).

The topic of backdoor-free pseudorandom generators was pioneered by Dodis et al. [DGG$^+$15]. They give a remarkable construction and security proof of a pseudorandom generator that can survive subversion. Their immunization procedure involves introducing some clean random bits beyond the control the adversary and so requires a more generous setting than our desired full subversion model. One of our results will provide similar security guarantees in the full subversion model, which does not require such clean randomness.

We first review the basic notions of PRG under subversion attacks and then provide a specific construction that mimics the classical Blum-Micali PRG construction in this kleptographic 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. We 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.

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

We adopt the definition from [DGG+15]: 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 PRG_q to denote the result of q iterations of PRG with outputs $r_1, \ldots, r_q$ (each $r_i \in \{0,1\}^{\ell'}$).

They then define the notion of a *backdoored PRG* [DGG+15]: 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 [DGG+15] 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 2.4, we remark that any PRG$^{\mathrm{A}}$ is a PRG$^{\mathrm{C}}$.

We first reformulate the definition of [DGG+15] 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, a 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 4.1. *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, for any* PPT *adversary* $\mathcal{A}$ *playing the following game (Fig. 6) with the challenger* $\mathcal{C}$*, there exists a watchdog* $\mathcal{W}$ *such that: 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*

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

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$^{\text{A}}$ *to stress that the public parameters are generated by the adversary.*

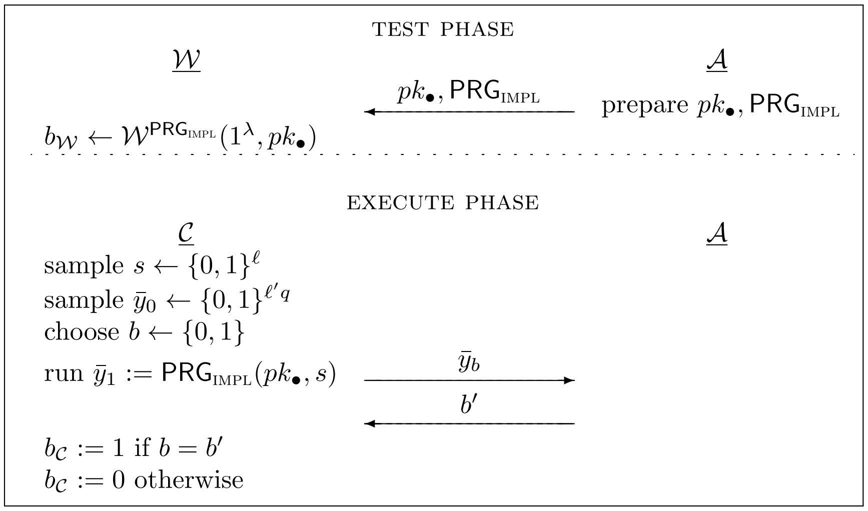
TEST PHASE
$\underline{\mathcal{W}}$
$\underline{\mathcal{A}}$
$pk_\bullet, \mathsf{PRG}_{\text{IMPL}}$
prepare $pk_\bullet, \mathsf{PRG}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{PRG}_{\text{IMPL}}}(1^\lambda, pk_\bullet)$
EXECUTE PHASE
$\underline{\mathcal{C}}$
$\underline{\mathcal{A}}$
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

Figure 6: Subversion-resistant$^{\text{A}}$ PRG Game

4.2 Constructing q-subversion-resistant$^{\text{A}}$ PRG 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 the KG implementation 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 [GL89] 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 [BM82] (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 $(s', b) \leftarrow \mathsf{PRG}(pk, s)$:

  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 appendix C

Lemma 4.2. *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 appendix B.1.

4.3 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 [DGG$^+$15]. However, we observe that this impossibility result only applies to an immunization procedure that operates on the *output* of the PRG$^{\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. [10]

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. PRG

See also the pictorial illustration for $\mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}$ in Fig 7.

The security analysis is very similar to that of Theorem 3.5; we defer the proof it to the full version.

[10]To interpret this results, the solution of [DGG$^+$15] 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.

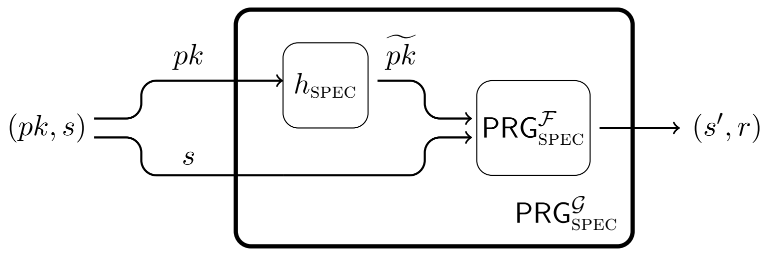
(pk, s)
pk
s
h_{SPEC}
$\widetilde{pk}$
$\mathsf{PRG}^{\mathcal{F}}_{\text{SPEC}}$
(s', r)
$\mathsf{PRG}^{\mathcal{G}}_{\text{SPEC}}$

Figure 7: Public immunization strategy for PRG.

Theorem 4.3. *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 4.4. *There are several points we would like to stress:*

- *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;*
- *All the results in this section can be applied to construct PRG$^{\text{C}}$ (instead of PRG$^{\text{A}}$). In that case, the adversary provides a stateless* KG *implementation, we can even work in the split-program model to apply our immunizing strategy to broader practical settings.*

5 Subversion-Resistant Signatures

In this section, we consider how to design digital signatures against the kleptographic attacks. Previously results [AMV15, BPR14] suggest that a unique signature scheme [GO93, Lys02] is secure against subversion on the signing algorithm assuming it satisfies the *verifiability* condition that every message signed by the sabotaged $\mathsf{Sign}_{\text{IMPL}}$ should be verified via $\mathsf{Verify}_{\text{SPEC}}$. The main question left is that in those constructions the key generation and verification algorithms have to be faithfully implemented, while in practice, all implementations normally come together. Note that all our results obtained in previous sections are actually in the *complete subversion* model. Here we will apply our subversion-resistant TDOWPs to bridge this gap, and construct the first subversion-resistant signature schemes in the more realistic complete subversion model.

We stress that whether a reduction between two primitives in the classical crypto world can be bring to clipto world turns out to be highly non-trivial. We will see that the well-known reduction for full domain hash can not go through in clipto setting when we try to build a subversion-resistant$^{\text{C}}$ signature from a TDOWP$^{\text{C}}$. (see remark 5.2 and the proof for Thm. 5.3 for more details).

Following our general framework, it is easy to derive a definition for subversion-resistant signature scheme. As pointed out in [AMV15], 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 obtain and check the transcripts generated during the game between $\mathcal{C}$ and $\mathcal{A}$.[11] Next we define the security for digital signature scheme in the

[11] Note that for digital signature schemes in practice, it is much more reasonable to assume an online watchdog than an omniscient watchdog as in [BPR14, DFP15]. Indeed, due to the nature of signature schemes, the transcripts consist of message-signature pairs, can be publicly verified, and an online watchdog is sufficient.

complete subversion model where all algorithms are implemented by the adversary, including the key generation algorithm.

Definition 5.1. *A signature scheme Π with specification $\Pi_{\text{SPEC}} = (\mathsf{KG}_{\text{SPEC}}, \mathsf{Sign}_{\text{SPEC}}, \mathsf{Verify}_{\text{SPEC}})$, is* **subversion-resistant**$^{\text{C}}$ *in the online watchdog model, if for any* PPT *adversary $\mathcal{A}$ playing the following game (Fig 8) with the challenger $\mathcal{C}$, there exists a watchdog $\mathcal{W}$, such that: 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*

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

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

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

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}.

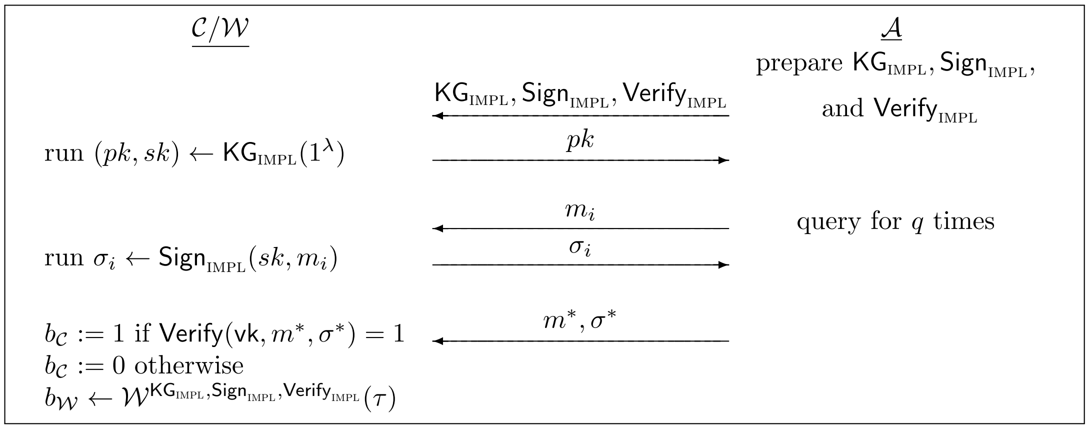
$\mathcal{C}/\mathcal{W}$
$\mathcal{A}$
prepare $\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}}$, and $\mathsf{Verify}_{\text{IMPL}}$
$\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}}, \mathsf{Verify}_{\text{IMPL}}$
run $(pk, sk) \leftarrow \mathsf{KG}_{\text{IMPL}}(1^\lambda)$
pk
m_i
query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

Figure 8: Subversion-resistant$^{\text{C}}$ Signature Game, where $\tau := (pk, \{m_i, \sigma_i\}_{i\in[q]}, m^*, \sigma^*)$

To upgrade the previous results to the complete subversion model, the main challenging part is to protect the randomized key generation algorithm against subversion attacks. While the attacks shown by Bellare et al [BPR14] are so devastating that the adversary can use a sabotaged randomized implementation to leak any secret bit by bit, we observe that the key generation algorithm will be run only once (as in the security definition). It leaves us the opportunity that if we suggest a better designed specification, the leaked information via the implementations is very limited, so that unforgeability of the signature scheme might still be preserved. Next, we will prove that a variant of the widely deployed full domain hash scheme [Cor00, BR96] 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. We further remark that when instantiating its key generation with our subversion-resistant TDOWP, this variant gives a subversion-resistant signature scheme.

Constructing signature schemes with an online watchdog. Given a subversion-resistant$^{\text{C}}$ TDOWP, 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, $\mathcal{M}$ 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 $(pk, sk) \leftarrow \mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}(\lambda)$, where $\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}$ is given by:

  Compute $(i, t_i) \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$, and set $pk := i$ and $sk := t_i$;

- Signature generation $\sigma \leftarrow \mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}(pk, sk, m)$, where $\mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}$ is given by:

  Upon receiving message m, compute $\tilde{m} = h_{\text{SPEC}}(pk, m)$, and $\sigma = \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}(pk, sk, \tilde{m})$, where $pk = i, sk = t_i$.

- Verification algorithm $b \leftarrow \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}(pk, m, \sigma)$, where $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}$ is given by:

  Upon receiving message-signature pair (m, σ), if $\mathsf{Eval}^{\mathcal{SS}}_{\text{SPEC}}(pk, \sigma) = h_{\text{SPEC}}(pk, m)$ then set $b := 1$, otherwise set $b := 0$; here $pk = i$.

Remark 5.2. *We emphasize that using the full domain hash directly without inputting pk into the hash, it is possible that the random oracle query for m^* is asked before the implementations are prepared, in this case, the simulator has not received y^* yet, thus has no way to embed y^* into the target.*

Theorem 5.3. *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.*

Proof. Assume that $\mathcal{SS}$ is subvertible by a PPT adversary $\mathcal{A}_{\mathcal{SS}}$, we have the following: (1) the detection probability **Det** is negligible; (2) the advantage **Adv** is non-negligible δ. This means the implementations provided by $\mathcal{A}_{\mathcal{SS}}$ will be accepted by all online watchdog who will see the transcripts generated in the game; in addition $\mathcal{A}_{\mathcal{SS}}$ outputs a valid new forgery with non-negligible probability, say δ. We will next construct an adversary $\mathcal{A}_{\mathcal{F}}$ that can break the security of $\mathcal{F}_{\text{SPEC}}$.

Construction of $\mathcal{A}_{\mathcal{F}}$. $\mathcal{A}_{\mathcal{F}}$ simulates a copy of $\mathcal{A}_{\mathcal{SS}}$. In addition $\mathcal{A}_{\mathcal{F}}$ simulates the subversion-resistant$^{\text{C}}$ security game in the online watchdog model with $\mathcal{A}_{\mathcal{SS}}$.

Suppose $m_{1,1}, \ldots, m_{q_1,1}$ are the random oracle queries that $\mathcal{A}_{\mathcal{SS}}$ makes before outputting the implementations, $\mathcal{A}_{\mathcal{F}}$ answers all those using uniform strings.

Upon receiving the implementations of $(\mathsf{KG}^{\mathcal{SS}}_{\text{IMPL}}, \mathsf{Sign}^{\mathcal{SS}}_{\text{IMPL}}, \mathsf{Verify}^{\mathcal{SS}}_{\text{IMPL}})$ from $\mathcal{A}_{\mathcal{SS}}$, $\mathcal{A}_{\mathcal{F}}$ prepares faithful implementations of $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}$ and sends them together with $\mathsf{KG}^{\mathcal{SS}}_{\text{IMPL}}$ to the $\mathcal{F}_{\text{SPEC}}$ watchdog $\mathcal{W}_{\mathcal{F}}$. It is easy to see that $\mathcal{W}_{\mathcal{F}}$ will not complain since all PPT offline watchdogs accept $\mathsf{KG}^{\mathcal{SS}}_{\text{IMPL}}$ (without even need the transcripts).

Upon receiving i^*, y^* from the $\mathcal{F}_{\text{SPEC}}$ challenger $\mathcal{C}_{\mathcal{F}}$, $\mathcal{A}_{\mathcal{F}}$ directly forwards i^* as the public key to $\mathcal{A}_{\mathcal{SS}}$. Note that i^* comes exactly from $\mathsf{KG}^{\mathcal{F}}_{\text{IMPL}} = \mathsf{KG}^{\mathcal{SS}}_{\text{IMPL}}$.

Now if $\mathcal{A}_{\mathcal{SS}}$ makes another set of random oracle queries for $pk||m_{1,2}, \ldots, pk||m_{q_2,2}$, $\mathcal{A}_{\mathcal{F}}$ chooses a random index $t \in [q_2]$ and answers y^* as $h_{\text{SPEC}}(pk||m_{t,2})$, and randomly chooses $\{\sigma_j\}_{j \neq t}$ and returns $\{\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(pk, \sigma_j)\}_{j \neq t}$ as answers for all other queries $\{pk||m_{j,2}\}_{j \neq t}$. $\mathcal{A}_{\mathcal{F}}$ keeps a list for the values of all those queries. (We remark that without loss of generality here we assume all the prefixes are pk.)

If $\mathcal{A}_{\mathcal{SS}}$ then makes signing queries on $m_{1,3}, \ldots, m_{q_3,3}$, $\mathcal{A}_{\mathcal{F}}$ can simply find the corresponding $\{\sigma_{i,3}\}$ from the list and return them as the signatures. If $\mathcal{A}_{\mathcal{SS}}$ outputs a forgery m^*, σ^* and $m_{t,2} = m^*$, $\mathcal{A}_{\mathcal{F}}$ returns σ^* to $\mathcal{C}_{\mathcal{F}}$ as the pre-image for y^*.

Probabilistic analysis. Now we bound the success probability of $\mathcal{A}_{\mathcal{F}}$. Let $\delta' = \Pr[\sigma^* = x^*]$, where x^* is the input chosen by $\mathcal{C}_{\mathcal{F}}$. It is easy to see that $\delta' \geq \Pr[\sigma^* = x^* | m_{t,2} = m^*] \Pr[m_{t,2} = m^*]$.

Following condition (1), there exists a watchdog that checks (m^*, σ^*) using $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}$. We can claim that $\mathcal{A}_{\mathcal{SS}}$ made a random oracle query for $pk||m^*$; if not, the probability that $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(pk, \sigma^*)$ hits a completely random value is negligible.

With the above claim, $pk||m^*$ will appear in the adversaries' random oracle queries. Observe that pk is output by $\mathsf{KG}^{\mathcal{SS}}_{\text{IMPL}}$, and we can see that with overwhelming probability, $pk||m^*$ will appear in the random oracle queries after pk is provided. It follows that: $\Pr[m_{t,2} = m^*] \geq \frac{1}{q_2}(1 - \mathsf{negl}(\lambda))$.

Conditioned on $m_{t,2} = m^*$, from $\mathcal{A}_{\mathcal{SS}}$'s view, in the subversion-resistant$^{\text{C}}$ game, she is supposed to receive $\sigma_1, \ldots, \sigma_q$ that are output by $\mathsf{Sign}^{\mathcal{SS}}_{\text{IMPL}}$ on inputting the corresponding sk and messages. With the offline watchdog, and the property of the full domain hash, that those signatures will pass $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}$, i.e., $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(\sigma_i) = h_{\text{SPEC}}(m_i)$. This implies that the distribution of σ_i is the same from that comes from $\mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}$ by calling $\mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}(h_{\text{SPEC}}(m_i))$, which is identically distributed as how $\mathcal{A}_{\mathcal{F}}$ simulates the answers for the signing queries.Now together with condition (2), $\mathcal{A}_{\mathcal{SS}}$ will output a valid forgery w.r.t $\mathsf{Verify}^{\mathcal{SS}}_{\text{IMPL}}$. With the offline watchdog, $\mathsf{Verify}^{\mathcal{SS}}_{\text{IMPL}}(m^*, \sigma^*) = \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}$, i.e., $\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(\sigma^*) = h_{\text{SPEC}}(m^*) = y$, thus $\Pr[\sigma^* = x^* | m^* = m_{t,2}] \geq \delta(1 - \mathsf{negl}(\lambda))$.

Combing all above, we see that $\delta' \geq \delta / q_2$ which is non-negligible. We can see that $\mathcal{A}_{\mathcal{F}}$ breaks the security of $\mathcal{F}_{\text{SPEC}}$. This means our assumption is wrong, which completes the proof. □

Remark about the general feasibility of TDOWP$^{\text{C}}$. Our construction is based on TDOWP$^{\text{C}}$. We have already constructed a concrete scheme for subversion-resistant TDOWP in the split program model. This construction is applicable to broader practical settings and it may be used to instantiate our signature construction in the split program model. An alternative approach to constructing TDOWP$^{\text{C}}$ is to follow the construction idea for OWP$^{\text{C}}$. (Note that the OWP$^{\text{A}}$ construction in the previous section directly give us a OWP$^{\text{C}}$ construction.) We remark that the function name of TDOWP consists of a index-trapdoor pair (i, t_i), if we directly follow the OWP$^{\text{C}}$ construction idea, we need to assume the hash function can map from a pair to another pair. Alternatively, we may apply the hash to the trapdoor, and in addition we assume there is a public procedure that can compute the public index from its trapdoor. Due to lack of space, we omit the discussion of this and defer the details to the full version.

References

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 364–375, 2015.

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, April 2015.

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1431–1440, 2015.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, August 2014.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.

[CNE+14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 319–335, 2014.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000.

[DFP15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 579–598. Springer, Heidelberg, March 2015.

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, April 2015.

[DMSD15] Yevgeniy Dodis, Ilya Mironov, and Noah 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.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

[GO93] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, August 1993.

[HH09] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 202–219. Springer, Heidelberg, March 2009.

[HLv02] Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 77–92. Springer, Heidelberg, August 2002.

[JG02] Ari Juels and Jorge Guajardo. RSA key generation with verifiable randomness. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 357–374. Springer, Heidelberg, February 2002.

[LPS13] Jeff Larson, Nicole Perlroth, and Scott Shane. Revealed: The NSA's secret campaign to crack, undermine internet security. Pro-Publica, 2013. http://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, August 2002.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, April 2015.

[NIS12] 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.

[PLS13] Nicole Perlroth, Jeff Larson, and Scott 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.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[Sim86] Gustavus J. Simmons. A secure subliminal channel (?). In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 33–41. Springer, Heidelberg, August 1986.

[YY96] Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, August 1996.

[YY97] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.

A Omitted Definitions

Definition A.1. *A trapdoor OWP family $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ with specification* $(\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}}, \mathsf{Inv}_{\text{SPEC}})$*, is* **subversion-resistant**$^{\text{C}}$ *in the offline watchdog model, if for any* PPT *adversary $\mathcal{A}$ playing with the challenger $\mathcal{C}$ in the following game, (Fig 9), there exists a watchdog $\mathcal{W}$, such that: 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*

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

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

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

For convenience, we also say that such $\mathcal{F}_{\text{SPEC}}$ is a TDOWP^C in the offline watchdog model.

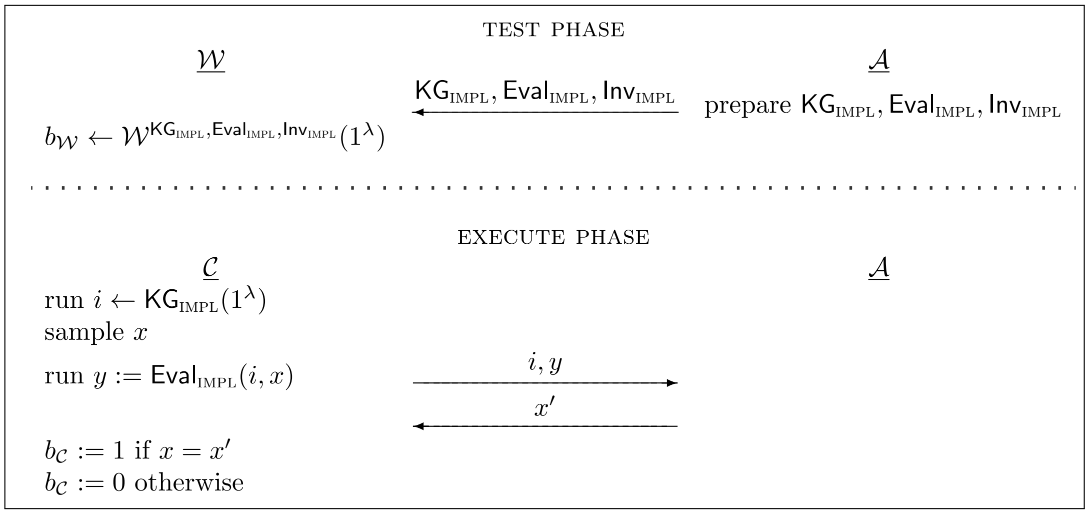
TEST PHASE
$\mathcal{W}$
$\mathcal{A}$
$\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}, \mathsf{Inv}_{\text{IMPL}}$
prepare $\mathsf{KG}_{\text{IMPL}}, \mathsf{Eval}_{\text{IMPL}}, \mathsf{Inv}_{\text{IMPL}}$
$b_{\mathcal{W}} \leftarrow \mathcal{W}^{\mathsf{KG}_{\text{IMPL}},\mathsf{Eval}_{\text{IMPL}},\mathsf{Inv}_{\text{IMPL}}}(1^\lambda)$
EXECUTE PHASE
$\mathcal{C}$
$\mathcal{A}$
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

Figure 9: Subversion-resistant$^{\text{C}}$ TDOWP Game

Definition A.2. *A function family $\mathcal{F}$ is trapdoor one way in the split-program model if there exist a pair of algorithms* $(\mathsf{RG}, \mathsf{dKG}, \mathsf{Eval}, \mathsf{Inv})$ *where (i.)* RG*, given a security parameter λ, outputs a uniform $\ell(\lambda)$-bit string r; (ii.)* dKG *is deterministic: given the randomness r it outputs a function index and trapdoor pair i, t; and (iii.) $\mathcal{F}$ is one-way under this procedure for generating* $(i, t) \leftarrow \mathsf{dKG}(r) : r \leftarrow \mathsf{RG}$*; (4).* $\forall x, \mathsf{Inv}(t, \mathsf{Eval}(i, x)) = x$.

B Omitted Constructions

B.1 Full Fledged Subversion Resistant PRG$^{\text{A}}$

We now extend our basic construction via iteration to show that the full-fledged Blum-Micali PRG construction, using a subversion-resistant$^{\text{A}}$ OWP, achieves a q-subversion-resistant$^{\text{A}}$ PRG for any

$q = \text{poly}(\lambda)$. Given a subversion-resistant$^{\text{A}}$ OWP $\mathcal{F}$ with its specification $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}})$, our full-fledged construction with its specification $\mathcal{H}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{H}}_{\text{SPEC}}, \mathsf{PRG}^{\mathcal{H}}_{\text{SPEC}})$ [12] is as follows:

- Parameter generation algorithm $pk \leftarrow \mathsf{KG}^{\mathcal{H}}_{\text{SPEC}}(\lambda)$:
  compute $i \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$ and set $pk := i$;
- Bit string generation algorithm $(s', r) \leftarrow \mathsf{PRG}^{\mathcal{H}}_{\text{SPEC}}(pk, s)$:
  upon receiving s and pk where $pk = i$, $s = s_1||s_2$, and $|s_1| = |s_2| = \ell$, compute the following:
  - let $s_1^0 := s_1$ and $s_2^0 := s_2$;
  - for $j = 1, \ldots \ell'$,
    $b_j := \langle s_1^{j-1}, s_2^{j-1} \rangle$;
    $s_1^j := s_1^{j-1}$; $s_2^j := \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i, s_2^{j-1})$; $s^j := s_1^j||s_2^j$;
  - $s' = s^{\ell'} = s_1||s_2^{\ell'}$; and $r = b_1 \ldots b_{\ell'}$.

  Please see also Figure 10 for pictorial illustration for implementations.

$$\begin{tikzcd}
s_1||s_2 \arrow[r] & s_1||f_i(s_2) \arrow[r] \arrow[d, "\langle s_1, s_2\rangle"] & s_1||f_i^2(s_2) \arrow[r] \arrow[d, "\langle s_1, f_i(s_2)\rangle"] & \ldots \arrow[r] & s_1||f_i^{\ell'}(s_2) \arrow[r] \arrow[d, "\langle s_1, f_i^{\ell'-1}(s_2)\rangle"] & \ldots \\
 & b_1 & b_2 & \ldots & b_{\ell'} &
\end{tikzcd}$$

Figure 10: One iteration of BM-PRG, and $f_i(x) := \mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}(i, x)$

Theorem B.1. *The full fledged construction above with specification $\mathcal{H}_{\text{SPEC}}$ is q-subversion-resistant$^{\text{A}}$ (for any polynomially large q), if $\mathcal{F}$ with specification $\mathcal{F}_{\text{SPEC}}$ is a subversion-resistant$^{\text{A}}$ OWP.*

Proof sketch. Following Lemma 4.2, $s_1||f_i(s_2)||b_1$ is pseudorandom, i.e., it is indistinguishable from "$u_1, \ldots, u_{2\ell}, v_1$", even to an adversary $\mathcal{A}$ who may set the public parameter i, where $\{u_t\}_{t\in[\ell]}, v_1$ are all random bits, and $f_i(s_2) = \mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}(i, s_2)$.

Observe that b_2 can be computed from $s_1, f_i(s_2)$; it follows that the adversary $\mathcal{A}$ (who has the backdoor) can not predict b_2 from b_1, otherwise she trivially distinguishes $s_1||f_i^1(s_2)||b_1$ from random, simply by computing b_2 from $s_1||f_i^1(s_2)$ and predicting using b_1 to see whether these are consistent. Similarly, $\mathcal{A}$ cannot predict b_3 from b_1, b_2: To see this, first observe that $\mathcal{A}$ can not predict b_3 from b_2, thus $\mathcal{A}$ can not predict b_3 from v_1, b_2 where v_1 is a random bit. If b_3 is predictable by $\mathcal{A}$ from b_1, b_2, then starting from $s_1||f_i(s_2)$, $\mathcal{A}$ computes b_2, b_3, and simply uses b_1 to test whether the predication is correct, so that she can distinguish $s_1||f_i(s_2)||b_1$ from $s_1||f_i(s_2)||v_1$, and further from $u_1 \ldots, u_{2\ell}, v_1$.

The above argument can be applied to any $j \in [\ell']$, so that $\mathcal{A}$ can not predict b_{j+1} from $b_1, \ldots, b_j$. Then—following the classic reduction from pseudorandomness to next-bit unpredictability—we can conclude that $b_1 \ldots b_{\ell'}$ is indistinguishable from uniform bits $\{0,1\}^{\ell'}$, even to $\mathcal{A}$. (This can be shown via the standard hybrid argument.) Last, inductively, we can conclude that $r_1, \ldots, r_q$ are indistinguishable from $\ell' \cdot q$ uniform bits. □

[12] PRG_q can be defined in a straightforward manner that runs the above PRG for q iterations, each iteration outputs ℓ' bits and updates the state for next iteration.

C Omitted Proofs

Proof of Lemma 3.2. Consider a TDOWP $\mathcal{F} = \{f_i\}$ with the associated specifications $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}})$. Assuming the trapdoors can be represented using $\ell(\lambda)$ bits, we construct a subvertible OWP family $\mathcal{G}$ with specification $\mathcal{G}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{G}}_{\text{SPEC}})$ as follows:

- Function generation $(i, r) \leftarrow \mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$, where $\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$ is given by:

  Run the $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$ algorithm and receive a function index/trapdoor pair (i, t_i); then discard t_i, and sample randomly $r \leftarrow \{0,1\}^{\ell(\lambda)}$; It outputs (i, r).

- Function evaluation $y \leftarrow \mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}(i, r, x)$, where $\mathsf{Eval}^{\mathcal{G}}_{\text{SPEC}}$ is given by:

  Upon inputting i, r, x, discard r, compute $y \leftarrow \mathsf{Eval}\text{SPEC}^{\mathcal{F}}(i, x)$; it outputs y.

- Invert function $\mathsf{Inv}^{\mathcal{G}}_{\text{SPEC}}$ is the same as $\mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}$.

It is easy to see that $\mathcal{G}$ is one way because $\mathcal{F}$ is one way (without t_i) in the classical setting.

While in the cliptographic setting, an adversary can first chooses a random key k for a symmetric key encryption scheme $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$. Then she provides a implementations below that can evade the detection and breaks the one-way security of the functions generated by the implementation:

- Function generation implementation, $(i, \tilde{r}) \leftarrow \mathsf{KG}^{\mathcal{G}}_{\text{IMPL}}(k)$, where $\mathsf{KG}^{\mathcal{G}}_{\text{IMPL}}(k)$ is given by:

  It first runs $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$, and receives an index i together with the corresponding trapdoor t_i; $\tilde{r} = \mathsf{SE.Enc}(k, t_i)$, i.e., $\tilde{r}$ is generated by encrypting t_i using k. It outputs $(i, \tilde{r})$.

- $\mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$ and $\mathsf{Inv}^{\mathcal{F}}_{\text{IMPL}}$ are the same as the specifications.

It is easy to see that when adversary obtains $i, \tilde{r}$, and $y = \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(i, x)$ she can simply decrypts $\tilde{r}$ to get $t_i = \mathsf{SE.Dec}(k, \tilde{r})$, and then inverts y to get x.

Furthermore, since $\mathsf{SE.Enc}$ is modeled as a which is assumed to be a pseudorandom permutation (PRP). PRP, the distributions of (i, r) returned by $\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$ and $(i, \tilde{r})$ returned by $\mathsf{KG}^{\mathcal{G}}_{\text{IMPL}}(k)$ for any k are computationally indistinguishable. □

Proof for Lemma: 4.2. The specification $\mathsf{KG}^{\mathcal{G}}_{\text{SPEC}}$ of the simplified Blum-Micali PRG outputs a random function index from the index set (by simply running $\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}$).

It is easy to see that the OWP function family $\overline{\mathcal{F}}$ given by $\mathsf{Eval}^{\overline{\mathcal{F}}}_{\text{SPEC}}(i, x_1||x_2) := x_1||\mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}(x_2)$, is subversion-resistant$^{\text{A}}$ if $\mathcal{F}$ is subversion-resistant$^{\text{A}}$. If the above basic construction is not subversion-resistant$^{\text{A}}$, then there exists a PPT adversary $\mathcal{A}$ such that (i.) all watchdogs will accept the public parameter pk and the implementation $\mathsf{PRG}^{\mathcal{G}}_{\text{IMPL}}$, and (ii.) $\mathcal{A}$ distinguishes the PRG output from a random $2\ell + 1$-bit string with some non-negligible probability δ. To expand condition (ii.), $\mathcal{A}$ can distinguish $s_1||f_i(s_2)||B(s)$ (where $B(s) = \langle s_1, s_2 \rangle$) from a uniform $(2\ell + 1)$-bit string for an uniform s.

Now from $\mathcal{A}$, we construct an adversary $\mathcal{A}_{\mathcal{F}}$ that breaks the subversion-resistance of $\mathcal{F}_{\text{SPEC}}$, as an OWP$^{\text{A}}$: $\mathcal{A}_{\mathcal{F}}$ runs $\mathcal{A}$ to get $pk, \mathsf{PRG}^{\mathcal{G}}_{\text{IMPL}}$. $\mathcal{A}_{\mathcal{F}}$ then implements $\mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$ as follows: when evaluating on inputs pk, x, $\mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$ calls $\mathsf{PRG}^{\mathcal{G}}_{\text{IMPL}}$ using $pk, \bar{0}||x$ and receives $\bar{0}||y||0$, $\mathsf{Eval}^{\mathcal{G}}_{\text{IMPL}}$ then discards the first ℓ bits and the last bit, and returns y. We can see that $\mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$ and $\mathsf{PRG}^{\mathcal{G}}_{\text{IMPL}}$ have identical input/output behavior; thus for any watchdog, if it accepts $pk, \mathsf{PRG}^{\mathcal{G}}_{\text{IMPL}}$, it also accepts $pk, \mathsf{Eval}^{\mathcal{F}}_{\text{IMPL}}$.

$\mathcal{A}_{\mathcal{F}}$ then continues the simulation: when receiving a challenge y, $\mathcal{A}_{\mathcal{F}}$ randomly samples r, and a bit b and sends $r||y||b$ for $\mathcal{A}$ to distinguish. If $b = B(s)$, $\mathcal{A}$ will output 1 with probability $1/2 + \delta$. It is easy to see that such adversary can predict the GL hardcore predicate B with advantage $\delta/2$. Following the GL proof [GL89], there exists another algorithm $I^{\mathcal{A}}$ that can invert y with probability $\delta' = \text{poly}(\delta/2)$.

$\mathcal{A}_{\mathcal{F}}$ runs $I^{\mathcal{A}}$ to invert y, and will output a correct pre-image with probability δ'. Combing both claims above, $\mathcal{A}_{\mathcal{F}}$ subverts $\mathcal{F}_{\text{SPEC}}$ with a non-negligible probability. This leads to a contradiction.

□