Destroying Steganography via Amalgamation: Kleptographically CPA Secure Public Key Encryption

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

May 29, 2016

Abstract

We describe a general technique to protect *randomized algorithms* against kleptographic attacks. We then apply the technique to construct the first IND-CPA secure public-key encryption scheme in the kleptographic setting. Our scheme preserves IND-CPA security, even when all relevant cryptographic algorithms—including key generation—are subject to adversarial subversion. The scheme requires no trusted parties or re-randomization reverse firewalls. The technique also gives a secure symmetric key encryption scheme that advances the state-of-the-art by permitting adversarial subversion of key generation and, furthermore, requiring no a priori decryptability assumptions.

Designing cryptographic primitives immune to kleptographic subversion is an active area which has led to remarkable new models and techniques; many of these are realizable by systems and can reduce the threat of such strong attacks. The feasibility of public-key encryption that is kleptographically secure in the CPA sense has been open till now.

1 Introduction

Modern cryptography has been spectacularly successful, leading to cryptographic tools with remarkable functionalities and security guarantees. Despite these advances, applying cryptographic tools to provide robust security *in practice* is a notorious challenge. In particular, practical settings often introduce threats that are not adequately reflected by conventional cryptographic security modeling. In this article, we focus on one such disparity between conventional modeling and practical application: the possibility of adversarial instantiation, subversion, or substitution of the cryptographic algorithms themselves.

One implicit assumption in typical cryptographic security modeling is that the deployed implementations of cryptographic algorithms actually realize their "official specifications." In practice, cryptographic implementations may diverge from their specifications for a variety of reasons, including unintentional programming bugs or malicious tampering; in the *kleptographic* setting, one considers the pathological possibility of *fully adversarial implementations* of cryptographic algorithms. The goal of such an adversary is to produce implementations of cryptographic algorithms

*University of Connecticut, acr@cse.uconn.edu

†Cornell University, qt44@cornell.edu

‡Columbia University, moti@cs.columbia.edu

§Virginia Commonwealth University, hszhou@vcu.edu

which compromise security while appearing to be correct even in the face of fairly intensive testing. (Formal models are discussed below.) We remark that the possibility of such kleptographic attacks arise whenever a "third-party" software library or hardware device is relied upon for cryptographic purposes.

The consequences of such attacks are rather surprising: It turns out that—in wide generality—*adversarial implementations of randomized algorithms may leak private information while producing output that is indistinguishable from the specification*. The possibility of such threats was showcased over two decades ago by Young and Yung [YY96, YY97]. Recently, starting with Bellare, Paterson, and Rogaway [BPR14], the topic has received renewed formal attention [DGG+15, BH15, MS15, DFP15, BJK15, AMV15, RTYZ15, DMSD15] motivated by startling evidence from the Snowden revelations of past deployment of kleptographic attacks. The topic was also notably highlighted by Rogaway's 2015 IACR Distinguished Lecture [Rog15] calling for community-wide efforts to explore defending mechanisms. One of the most striking recent discoveries [BPR14, BJK15] establishes that a steganographic channel can be embedded in the output of a subverted randomized (encryption) algorithm so that secret information can be exclusively leaked to the adversary. Such steganographic attacks can even be applied in settings where the subverted algorithms are stateless [BJK15, RTYZ15].

As mentioned above, the kleptographic setting features an adversary who may substitute malicious algorithms in place of intended cryptographic algorithms. The adversary's attack, however, is constrained by introducing a trusted party (we call it the "watchdog") who may *test* the (adversarially-provided) implementations against the specification—this testing can provide a measure of safety to the final users of the implementation. The major question is whether a combination of careful specification and testing can preserve security despite such a powerful adversary. The broad challenge is to rework the classical framework of cryptographic primitives and constructions so that they provide security in this new environment. Recent efforts have partially explored the landscape and clearly identified the critical role played by randomized algorithms in the kleptographic setting. Indeed, existing defending strategies roughly fall into two categories:

- "Abandon" randomized algorithms and turn to deterministic counterparts, e.g., use deterministic encryption with a unique ciphertext property as suggested in [BPR14, BH15, DFP15, BJK15] for encryption schemes. Of course, key generation is a sticking point in this approach, as it must be randomized; this has been handled by simply placing key generation outside the subversion model (i.e., assuming honest key generation).

- Use a trusted reverse firewall to re-randomize all incoming/outgoing communication generated by the randomized algorithms [MS15, DMSD15].

While insisting on deterministic algorithms has favorable properties, it necessarily places many central notions of security entirely out of reach: in particular, IND- security for public-key encryptions is unattainable. Furthermore, as mentioned above, the process of key generation is inherently stochastic; this difficulty has been avoided by placing key generation outside the subversion model (or, equivalently, assuming it to be trusted). On the other hand, the firewall model can provide general feasibility results but requires the assumption of an active trusted party (in particular, it requires a source of trusted randomness). As the large-scale goal of the study of kleptography is to reduce the need for trust in the underlying components, it is attractive to understand to what extent we can eliminate such trusted randomness, algorithmic elements, etc.

In this paper, we address the following central questions:

1. *Is it possible to generically annihilate subliminal channels in subverted randomized algorithms without a trusted party?*
2. *Is it possible to achieve IND-CPA secure public-key encryption without trust in any of the component algorithms.*

We provide affirmative answers to both questions.

Our principal technique (completely realizable in systems) involves a certain decomposition of algorithms into a few functional "components," which are tested by the watchdog and may be independently run and "re-assembled" by the user.

The kleptographic model, in brief. The kleptographic model is meant to capture a situation where an adversary (or "big brother" as we shall occasionally say) has the opportunity to implement (and, indeed, "mis-implement" or subvert) our basic cryptographic tools. On the other hand, the model also introduces a "watchdog" who will attempt to check, via black-box testing, that the cryptographic tools have been faithfully implemented by the adversary. We imagine the adversary to be "proud but malicious": the adversary wishes to interfere with security, but does not wish to be exposed as a fraud by the watchdog. The model, in brief:

1. **Specification (see also ⋆ below.)** The cryptographic primitive is *specified* as a tuple $\Pi_{\text{SPEC}} = (F^1_{\text{SPEC}}, \ldots, F^k_{\text{SPEC}})$ of "functionalities." Each F^i_{SPEC} is either a function $F : \{0,1\}^* \to \{0,1\}^*$ (which thus specifies a deterministic algorithm) or a family of probability distributions, so that $F(x)$ is a probability distribution over $\{0,1\}^*$ for each x (which thus specifies a randomized algorithm).

2. **Subversion.** The adversary provides us with all algorithmic and cryptographic building blocks; that is, the adversary provides us with an "implementation" $(F^1_{\text{IMPL}}, \ldots, F^k_{\text{IMPL}})$ for each of the functionalities F^i_{IMPL}. Observe that the adversary may provide even the algorithms that generate randomness and random objects such as keys. Of course, in general the implementations may disagree with the specification, which can provide the adversary a novel avenue to attack the primitive.

3. **Testing; the watchdog.** The algorithmic and cryptographic building blocks are then sent to trusted testing facility, the *watchdog*. The watchdog is aware of the official specification, and may query the adversary's implementations (treating them as block-boxes) in an attempt to detect disagreements between the implementations and the specifications.

4. **The security game.** Assuming that the watchdog is satisfied, the implementations are pressed into service, at which point their security is modeled by a conventional security game.

⋆ **Remark: Decomposition and Amalgamation.** We permit the designer of the cryptographic primitive (who determines its specification) an extra dimension of freedom which can assist the watchdog in his verification task: We permit the designer to functionally *decompose* the primitives into a fixed number of "pieces." For example, rather than specifying a function of interest $f : X \to Y$, the designer may instead specify two functions $h : X \to W$ and $g : W \to Y$ with the property that $f = g \circ h$. (Thus h and g together implicitly specify f.) An important example in our setting is specifying a randomized algorithm $\mathsf{G}(x)$ as a composition $\mathsf{dG}(x, \mathsf{RG}(1^k))$, where dG is a deterministic algorithm and RG is an algorithm which, given the

input 1^k, produces k uniformly random bits as output. In general, the decomposition may be arbitrary, but may only involve $O(1)$ pieces and cannot depend on input length.

Our contributions. In this paper, we develop general techniques that eliminate subliminal channels introduced by adversarial implementations. We apply this general technique to construct symmetric-key and public-key encryption schemes preserving the IND-CPA security when all the algorithms are subject to adversarial subversion. In particular,

1. We first define a property of "stego-freeness" by adjusting the previous models of [BPR14, RTYZ15] to characterize whether an implementation of an algorithm can be considered to be following its specification in the kleptographic setting (where there is a watchdog). The model includes several variants depending on the algorithm inputs and choices about the decomposition. We also define a notion of "subversion resistance" for encryption schemes by considering the kleptographic IND-CPA security game where the challengers use adversarial implementations and the implementations are interrogated by an "offline" watchdog. (See below for formal definitions.)

2. We then consider how to defend against steganographic channel attacks by the simple non-black-box technique of decomposition-and-amalgamation. We first extend the attacks of [BPR14, BJK15, DGG+15] to the setting where a public-key encryption algorithm is split into two components: randomness generation (even when the immunization function is modeled as a random oracle) and a deterministic component. We then demonstrate a "double-splitting" strategy in which randomness generation is carried out by two independent components $\mathsf{RG}_0, \mathsf{RG}_1$. We prove that when r_0, r_1 are sampled independently from RG_0 and RG_1, mixing them with an immunization function Φ can indeed destroy subliminal channels in the implementation of a wide class of randomized algorithms in the random oracle model.[1] We also consider how to achieve such results in the standard model (without a random oracle); see Sec. B in the appendix.

3. We further apply this general technique to immunize each algorithm of a symmetric-key (single-bit) encryption scheme, including key generation and encryption. Our construction preserves IND-CPA security of the underlying primitive. We then focus on constructing symmetric-key encryption schemes for large message spaces. To defend against input-trigger-style attacks [DFP15], we allow the user to carry out one single trusted addition. We also consider correctness in the kleptographic setting and draw connections to the theory of self-correcting programs. These techniques can be applied directly to immunize public-key encryption, and our construction gives the first IND-CPA secure scheme in the kleptographic setting without relying on a trusted party. Finally, we discuss some further applications, which include bypassing the impossibility results for publicly immunized *outputs* of a backdoored PRG; see Sec. C in the appendix.

Related works. Kleptography, as noted, was introduced by Young and Yung [YY96, YY97]; they primarily highlighted the possibility of subverting key generation, and left open the problem of defending against such subversion. Recent work [RTYZ15] has made initial progress on the problem of protecting key generation for specific cryptographic algorithms (trapdoor one-way

[1] Same as [RTYZ15], we only assume the specification $\mathsf{RG}_{\mathrm{SPEC}}$ to be an random oracle, while the implementation $\mathsf{RG}_{\mathrm{IMPL}}$ can be arbitrarily subverted.

permutations, pseudorandom generators, and digital signature scheme). However, these techniques are highly tuned to specific algorithms and do not remove arbitrary steganographic channels, which is one of our main goals.

Several research threads have studied the kleptographic setting, developing both new attacks and defending mechanisms. In particular, Bellare, Paterson, and Rogaway [BPR14] studied subverted randomized encryption algorithms, building a steganographic channel that leaks secrets bit by bit; they also developed defending mechanisms in the setting where key generation is honest. Such subliminal channel attacks turn out to be the major obstacle in this area, and have been further explored by Ateniese et al. [AMV15], Bellare et al. [BH15, BJK15], Degabriele et al. [DFP15], and Dodis et al. [DGG+15]. A common feature of these works [BPR14, BH15, BJK15, DFP15] is to adopt deterministic algorithms and to assume honest key generation. Additionally, these works do not rely merely on testing: Most, in fact, require an a priori "decryptability" condition which demands that every message encrypted using the implementation should be decrypted correctly using the specification. A notable exception is [DFP15]; however, they rely on a watchdog that possesses access to the actual challenger–adversary communication transcript (including the internal state of the challenger).

Other works [MS15, DMSD15] considered defending mechanisms with a "reverse firewall" that is trusted to generate good randomness and can "re-randomize" incoming and outgoing communication. This model is attractive as it may permit quite general feasibility results; on the other hand, it introduces another trusted party (and source of trusted randomness).

In contrast to previous work,our goal is to develop CPA-secure encryption in a much stricter model that does not require strong watchdogs, clean keys, trusted randomness, or decryptability assumptions.

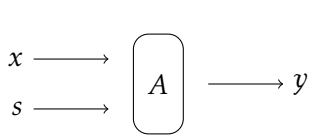
2 Definitions and Models

The adversary in kleptographic settings is *"proud-but-malicious"*: The adversary prefers subversion to be "under the radar" of any possible detection; on the other hand, she still wishes exploit her power of subversion to violate security. As explained in the introduction, our central focus will be the challenge of *generically destroying* subliminal channels which may have been adversarially embedded in a subverted algorithm. We briefly recall the notion to set the stage for the basic definitions.

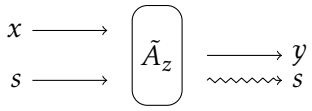
Consider an (honest) randomized algorithm A which takes an input x and has additional access to "secret" bit $s \in \{0,1\}$. The algorithm produces a random output y, which we assume leaks no information about s. A fundamental result in steganography [Sim83, Sim86, HLv02] asserts that is is possible to construct a subverted algorithm $\tilde{A}_z$, whose behavior is determined by a hidden random string z, so that

- for all inputs x and s, the distribution produced by $\tilde{A}_z(x,s)$ (including the random selection of z) is *identical* to the distribution produced by $A(x,s)$, and hence leaks no information about s; but,
- with knowledge of z, the output of $\tilde{A}_z$ is highly correlated with s. In particular, an adversary with access to z can use the output of $\tilde{A}_z$ to infer s with high probability.

See Figs. 1a and 1b.



(a) A randomized algorithm which leaks no information about s.



(b) A subverted algorithm depending on a hidden random string z. Its output (y) is indistinguishable from A, but with knowledge of z it leaks s.

Figure 1: Embedding a subliminal channel in a randomized algorithm A.

As mentioned above, subliminal channels are a major security obstacle in the kleptographic setting and our main result is a method for destroying them. Intuitively, to achieve this goal, we demand that the adversary cannot learn any extra information from the output of a subverted implementation of an algorithm so long as it has passed the checking of the watchdog. We adopt the general kleptographic definitional framework of [RTYZ15], and generalize the notion defined by the surveillance game in [BPR14] to formulate our definition of "destroying a steganographic channel"—this defines a new notion that we call "stego-freeness." Stego-free specifications for algorithms will be the stepping stones to our final construction of cryptographic primitives (e.g., public-key encryption) with subversion resistance.

In the definitional framework of [RTYZ15], the adversary $\mathcal{A}$ provides subverted implementations of the cryptographic algorithms for a particular primitive; the challenger $\mathcal{C}$ must then play the (standard) cryptographic security game for the primitive with $\mathcal{A}$. Of course, the challenger uses the subverted implementations during the security game. However, to protect the challenger, there is a *watchdog* $\mathcal{W}$ who tests the subverted implementations, by comparing them with the specification of the algorithms. The adversary "wins" if she can manufacture implementations so that she can win the security game and—at the same time—evade the detection of $\mathcal{W}$. One can arrive at a variety of different definitions based on the order of quantification for $\mathcal{W}$ and $\mathcal{A}$, how exactly $\mathcal{W}$ is permitted to test the implementations, and whether $\mathcal{W}$ is given any further information (such as a transcript of the security game). We refer to [RTYZ15] for the precise details.

In this paper we will adopt the strongest of the definitions of [RTYZ15] (which gives the watchdog the least power): in their terminology, we will consider a *universal* and *offline* watchdog. In such a definition, the watchdog only tests the implementation once with only oracle access. In particular, $\mathcal{W}$ has no access to the actual communications during the security game. Moreover, the description of the watchdog is quantified before the adversary.[2] (Thus, security for a particular primitive requires that there is a single watchdog that can protect against all possible adversaries.)

To formalize the notion that the adversary cannot learn unintended information from an implementation, we adapt the surveillance game from [BPR14] (which was defined for symmetric key encryption): specifically, we compare the information leaked by the implementation with that leaked by the specification (or, equivalently, an honest implementation).

Defining stego-freeness. We now formally define stego-freeness for any (randomized) algorithm G under subversion. Following the basic kleptographic models described above, the adversary $\mathcal{A}$ prepares a (potentially subverted) implementation G_{IMPL} of the algorithm G; we let G_{SPEC} denote the specification of the algorithm. The goal of the adversary is to utilize G_{IMPL} to leak secret information

[2] This is stronger than most of the definitions in the literature. The closest one is [DFP15]; however, their watchdog has to take the transcript between $\mathcal{C}$ and $\mathcal{A}$ as inputs which implicitly implies the dependence of the running time on $\mathcal{A}$.

exclusively to her via the outputs that G_{IMPL} produces (as in the discussion above). Stego-freeness means **either** the adversary $\mathcal{A}$ cannot learn any extra information from the outputs of G_{IMPL} (in comparison with that of G_{SPEC}), **or** the subversion can be detected by the watchdog $\mathcal{W}$ (using oracle access to G_{IMPL})—this is characterized by the detection advantage $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ below. Depending on how communication is generated, we have a variety of definitions; we begin with the following elementary version.

Definition 2.1 (stego-free, basic form)**.** *Consider a (randomized) algorithm* G *with specification* G_{SPEC}*. We say such* G_{SPEC} *is* stego-free in the offline watchdog model *if there exists a* PPT *watchdog* $\mathcal{W}$ *so that for any* PPT *adversary* $\mathcal{A}$ *playing the following game (see Fig. 2), it satisfies that either,*

$$\mathbf{Adv}_{\mathcal{A}} \text{ is negligible,} \quad \text{or} \quad \mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible.}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr[b_{\mathcal{C}} = 1] - 1/2|$*, and,* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{G}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\Pr[\mathcal{W}^{\mathsf{G}_{\text{SPEC}}}(1^\lambda) = 1]\right|$

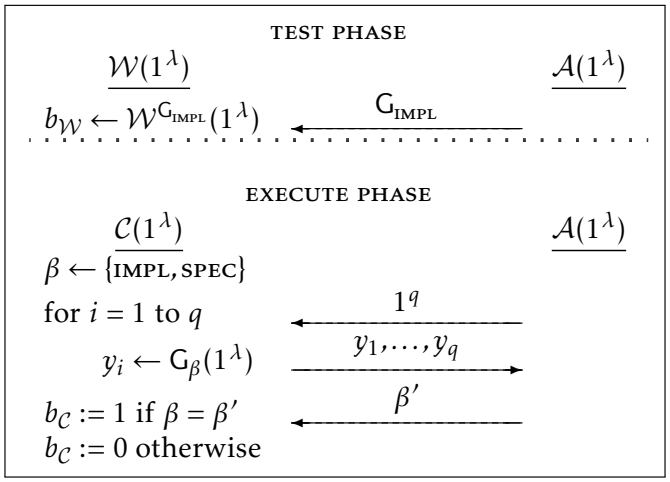


Figure 2: A game for stego-freeness.

Note that the definition requires only non-negligible detection probability on the part of the watchdog. We remark, however, that the detection probability can be directly amplified by repetition.[3]

We remark that an offline watchdog can ensure that the implementation of a *deterministic* algorithm disagrees with its specification with negligible probability when inputs are drawn from a public input distribution (which can be considered as an efficiently samplable source that the watchdog can sample inputs from it):

Lemma 2.2 ([RTYZ15])**.** *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 security parameter* λ*, public input distributions* $X^1_\lambda, \ldots, X^k_\lambda$ *are defined respectively. If* $\exists j \in [k], \Pr[F^j_{\text{IMPL}}(x) \neq$

[3] Trivial amplification transforms a gap of ϵ to $1 - \delta$ with $k = \epsilon^{-1} \log(\delta^{-1})$ repetitions. As the watchdog's running time is fixed independent of the adversary, however, amplification cannot be adapted to a particular non-negligible function. If the watchdog is permitted a number of samples that depends on the adversary, then one can amplify non-negligible detection probability to $1 - o(1)$ for an infinite set of inputs.

$F^j_{\text{SPEC}}(x) : x \leftarrow X^j_\lambda]$ *is non-negligible, this can be detected by a* PPT *offline watchdog with non-negligible probability.*

More general definitions of stego-freeness. In the above game, G only takes as input a fixed security parameter (often ignored later in the paper); this definition can capture algorithms like randomness generation and key generation when we instantiate G to be the corresponding functionality. Besides the security parameter, we can consider algorithms which take richer inputs. Such extensions will be important for our applications, and can significantly complicate the task of destroying an embedded steganographic channel.

First, we consider an input taken from a small (polynomial-size) input domain D. In this setting, the adversary can request evaluations of $\mathsf{G}_{\text{IMPL}}(1^\lambda, x)$ for values $x \in D$ of her choice. However, as D is small it suffices to analyze the "oblivious" version of the algorithm which simply responds with an evaluation of $\mathsf{G}_{\text{IMPL}}(1^\lambda, x)$ for *all* $x \in D$—this permits us to view this as a special case of the preceding "input-free" setting. (If, e.g., $D = \{0,1\}$, then the messages will simply be generated as $\{y_{i,0} \leftarrow \mathsf{G}_\beta(1^\lambda, 0), y_{i,1} \leftarrow \mathsf{G}_\beta(1^\lambda, 1)\}_{i\in[q]}$; see Figure 7 of Section 4.1 for a concrete example.)

Beyond the two previous cases, we may consider algorithms taking inputs from a large domain. The most straightforward adaptation permits the adversary to sample $\mathsf{G}_{\text{IMPL}}(1^\lambda, x_i)$ of her choosing. However, this model suffers from a crippling "input trigger" attack [DFP15] (where the adversary hides some secret information at a particular "trigger" location x which can be impossible for an offline watchdog to detect); we discuss this in detail later. However, there is a compromise setting that captures many cases of actual interest and permits strong feasibility results. In this setting we permit the randomized algorithm G_{SPEC} to be specified along with an *input generator* $\mathsf{IG}_{\text{SPEC}}$. The specified behavior of the algorithm is to produce $\mathsf{G}_{\text{SPEC}}(1^\lambda, \mathsf{IG}_{\text{SPEC}}(1^\lambda))$; note that both IG and G may be randomized, and can be subverted by the adversary.

Thus, along with G_{IMPL}, the adversary specifies an *input generator* IG that samples inputs (for G). The input generator may be an arbitrary PPT algorithm with the condition that given 1^λ it produces outputs of length exactly λ. Then challenges $\{y_i\}$ in the security game are generated by first sampling $m_i \leftarrow \mathsf{IG}(1^\lambda)$, and then obtaining $y_i \leftarrow \mathsf{G}_\beta(1^\lambda, m_i)$ by calling G_β using inputs 1^λ and m_i. Note that the adversary could use IG to produce some specific input "triggers" where G_{IMPL} deviates from G_{SPEC}. This more general notion of stego-freeness (with a "public" input distribution) captures algorithms that take the output of other algorithms as input, which will be critical when we reason about amalgamation of algorithms. See Figure 3 below for a unified game, where the algorithm may take both types of inputs.

Definition 2.3 (stego-free, general form)**.** *We say that a randomized algorithm* G *is stego-free if it satisfies Definition 2.1 with the security game of Figure 3 (in such cases we let the domain D be determined from context). Note that the* PPT *input generator* IG *may be determined by the adversary during the game.*

Which of the definitions (2.1 or 2.3) is appropriate for a given randomized algorithm can be determined from context, depending on whether an input generator is specified.

As mentioned above, an even stronger definition is obtained by permitting the adversary to simply choose the input m_i for each y_i directly. This notion reflects stego-freeness for algorithms with adversarially chosen inputs. Such a subverted implementation may have a hidden "trigger" that was randomly drawn during the (adversarial) manufacturing process and can permit the adversary to easily win the stego-freeness distinguishing game. In fact, such a trigger attack does not even require that G be randomized: for example, consider the algorithm $\mathsf{G}_{\text{SPEC}}(1^\lambda, x) := x$,

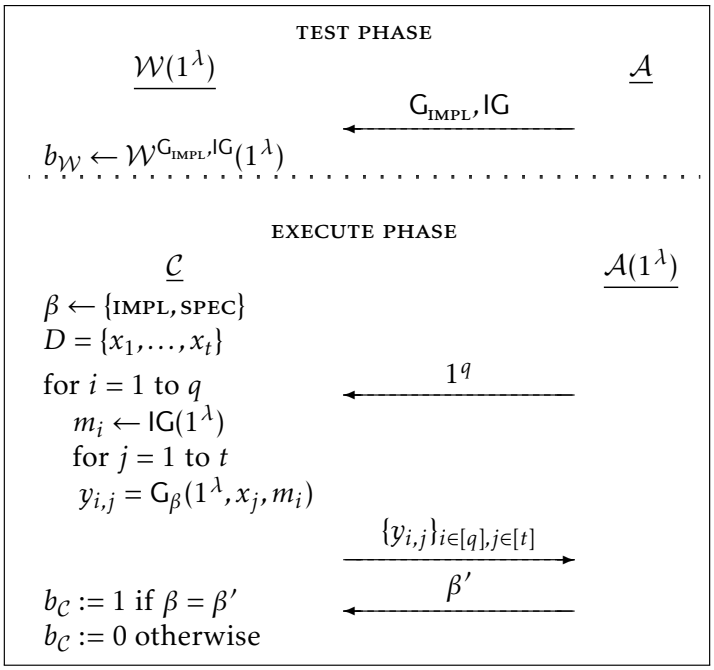


Figure 3: The stego-freeness game with input distribution $\{1^\lambda\} \times D \times \mathsf{IG}$.

defined for $x \in \{0,1\}^\lambda$. The adversary then uniformly draws $z \leftarrow \{0,1\}^\lambda$ and defines

$$\mathsf{G}_{\text{IMPL}}(1^\lambda, x) = \begin{cases} 0^\lambda & \text{if } x = z, \\ x & \text{otherwise.} \end{cases}$$

As the placement of the trigger (z) is random, the watchdog cannot detect disagreement between G_{IMPL} and G_{SPEC}, while the adversary can distinguish these algorithms easily by querying z. In a practical setting, an algorithm with such an input trigger can leak arbitrary private data to an adversary in a way undetectable to an offline watchdog. This was formally demonstrated in [DFP15] and called an "input-triggered subversion attack." Nevertheless, we will discuss in Section 4.1 a method for sidestepping this impossibility with only an offline watchdog by assuming some minimum trusted operations, such as "one trusted addition."[4]

These definitions of stego-freeness are sufficient for capturing most of the interesting use cases we will require (e.g., reflecting key generation and encryption).

Remark 2.4. *Following Lemma 2.2, observe that if G_{SPEC} is deterministic, an offline watchdog can ensure that inconsistencies ($G_{\text{SPEC}}(x) \neq G_{\text{IMPL}}(x)$) occur with only negligible probability when inputs are sampled from the input distribution* IG *(by drawing and testing a sample). In particular, deterministic algorithms with a public input distribution satisfy stego-freeness in a straightforward fashion.*

Discussions about stego-freeness and steganography. We emphasize two properties of these definitions. First, if a proposed specification satisfies such definitions, direct use of the implementation—rather than the specification—preserves the typical security guarantees originally possessed by the

[4] All previous works either simply assume it won't happen (the decryptability assumption) or employ an omniscient watchdog who has access to the transcript between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$, (and the secret key of $\mathcal{C}$).

specification. This enables us to provide fairly modular security proofs by designing specifications for each algorithm with stego-freeness.

The second, and more critical, issue pertains to the feasibility of achieving these notions of stego-freeness: in particular, at first glance they appear hopeless. It is known that general steganography is always possible over a channel with sufficient entropy [Sim83, Sim86, HLv02]. This implies that the subverted algorithm G_{IMPL} can always produce a sequence of messages that enable the adversary to retrieve secret data from the (public) outputs $y_1, \ldots, y_q$. In particular, as shown by Bellare, Paterson, Rogaway in the seminal result [BPR14], a subverted randomized encryption algorithm can generate ciphertexts so that the adversary can recover the secret key bit-by-bit from the sequence of ciphertexts. Moreover, the distribution of these subverted ciphertexts is statistically close to the natural, unsubverted ciphertext distribution. To make matters worse, such attacks can be launched even if the subverted implementations are stateless [BJK15]. As a simple example of such subversion in our setting, consider the algorithm $\mathsf{G}_{\text{SPEC}}(1^\lambda)$ which outputs a uniformly random element of $\{0,1\}^\lambda$. Consider then the subverted implementation $\mathsf{G}^z_{\text{IMPL}}(1^\lambda)$ whose behavior is determined by a uniformly random string $z \leftarrow \{0,1\}^\lambda$ chosen by the adversary: the algorithm $\mathsf{G}^z_{\text{IMPL}}(1^\lambda)$ outputs a uniformly random element of the set

$$H = \{w \in \{0,1\}^\lambda \mid \mathsf{lsb}(F_z(w)) = 0\},$$

where $\mathsf{lsb}(x)$ denotes the least-significant bit of x and $F_z(\cdot)$ denotes a pseudorandom function (PRF) with key z. (Note that elements of H can be drawn by rejection sampling.) Of course, it is easy for the adversary to distinguish G_{IMPL} from G_{SPEC} (as G_{IMPL} only outputs strings with a particular property that is easily testable by the adversary who has z). On the other hand, no watchdog can distinguish these algorithms without breaking the PRF. This suggests that if the user makes only *black-box* use of the subverted implementation of randomized algorithms, it is hopeless to achieve stego-freeness. This motivates the following *non-black-box* model.

The split-program methodology and trusted amalgamation. To overcome the steganographic attacks discussed above, we propose a slightly modified model which permits the specification of an algorithm to be split into several components. In this *split-program model*, each component of the implementation is exposed to the watchdog to check, while the challenger will amalgamate the components to yield the fully functional implementation. Of course, the implementation of each component is still presented by the adversary. We permit decomposition into only a constant number of components (independent of input length), with the demand that the desired algorithm can furthermore be expressed as the composition of a constant number of the components. Note that such a "split-program" presentation of an algorithm rules out any gate-by-gate treatment, as the model permits only a constant number of compositions. Intuitively, this simple non-black-box presentation of a randomized algorithm not only provides more opportunity to enforce the malicious implementation to follow a certain pattern, but also enables the watchdog to do more delicate checking on the inner structure.

One example of this framework is the simple split-program method proposed in [RTYZ15] to study certain randomized algorithms: they begin by specifying a (general) randomized algorithm G as a pair $(\mathsf{RG}, \mathsf{dG})$ where RG is the *randomness generation* algorithm, responsible for generating a uniform random string of appropriate length, and dG is a deterministic algorithm that takes the input to the original randomized algorithm G and the random coins produced by RG to produce the final output. They then add to this specification a third deterministic algorithm Φ which acts as a kind of "immunization function" for the random bits generated by RG. Specifically, given the

implementations ($\mathsf{RG}_{\text{IMPL}}$, $\mathsf{dG}_{\text{IMPL}}$, Φ_{IMPL}), the challenger *amalgamates* them by first querying $r_0 \leftarrow \mathsf{RG}_{\text{IMPL}}$, "sanitizing" this randomness by passing it into Φ_{IMPL} to receive $r \leftarrow \Phi_{\text{IMPL}}(r_0)$ and, finally, running $y \leftarrow \mathsf{dG}_{\text{IMPL}}(r)$. They show that in several contexts such an "immunization" can preserve security even under subversion. We remark that a simple decomposition and amalgamation of this form cannot destroy steganography in general, and we show an explicit attack in this model; see Sec. 3.1.

To reflect such trusted amalgamation in our security model, we permit the challenger to carry out (a constant number of) compositions without molestation; that is, the notion of "composition" is protected from adversarial subversion. This is implicit in the security games defined, e.g., in Figure 12, and in Figure 13, in the appendix.

As mentioned above, the split-program method proposed in [RTYZ15] permitted them to establish security (in the kleptographic model) for specific cryptographic primitives. In this paper, we will show that this general methodology has remarkable power against subversion: by further decomposition and amalgamation, we show it is possible to generically destroy steganographic channels. This provides us a family of tools for developing kleptographically-secure cryptography without abandoning randomized algorithms.

Stateful algorithms. As discussed above, steganographic attacks can be launched even if the implementation is stateless. To simply our presentation, most of our discussion adopts this stateless assumption. However, adaptations of our techniques can provide security even for stateful implementations (in the sense that each functionality maintains internal state). These amplified results require a slightly stronger watchdog (whose running time can depend on the adverary) and more detailed control of the subverted algorithm to ensure that they receive inputs from public distributions. See Remark 3.3 in Sec. 3.2 for more discussion.

3 Eliminating Subliminal Channels in Randomized Algorithms

In this section, we will present our main result: provable destruction of any subliminal channels in subverted implementations of randomized algorithms.

First, we motivate our new constructions by showing that the steganographic attacks of [BPR14, BJK15] can still be carried out in the simple split-program model introduced by [RTYZ15]. The attack succeeds even if the associated "immunizing function" Φ is a trusted hash function modeled as a random oracle. This indicates that some stronger form of immunization is necessary for destroying steganography.

Next, we present our main result in Section 3.2: a generic transformation that destroys steganography in randomized algorithms. The basic technique relies on "double-splitting" the randomness generating procedure coupled with a public immunizing function: specifically, randomness generation is expressed as two algorithms $\mathsf{RG}_0, \mathsf{RG}_1$ in conjunction with a public immunizing function Φ. We prove that with this "extra" split, when the outputs $r_0 \leftarrow \mathsf{RG}_0, r_1 \leftarrow \mathsf{RG}_1$ are independently sampled from RG_i and mixed with Φ, the new specification of the randomness generation algorithm is indeed stego-free in the random oracle model. Note that all components, including the immunizing function Φ, are subject to subversion. With randomness generation cleaned in this way, we can further destroy the subliminal channel in a large class of randomized algorithms. These results enable us to overcome the major obstacle in designing cryptographic specifications that satisfy subversion resistance.

Transition to the standard model. Finally, we consider how to achieve stego-freeness without random oracle. The main observation is that the watchdog essentially guarantees that each copy of $\mathsf{RG}^i_{\text{IMPL}}$ can provide at least $\log n$ bits entropy, if we are willing to have more components for randomness generation, we can accumulate entropy using some simple immunizing function (and stretch it using a PRG). Due to lack of space, for details of standard model constructions, we defer to Sec. B in the appendix.

3.1 Impossibility of publicly immunizing a single random source

Previous works [BPR14, BJK15] demonstrated that if a (subverted) randomized algorithm is used in a black-box fashion, a subliminal channel can always be embedded in its output distribution. Here we point out that a similar attack exists even if we adopt the techniques described in [RTYZ15] which (i.) split the algorithm G into RG (which generates randomness) and dG (which is deterministic), (ii.) introduce Φ—an "immunizing" function—and, ultimately, (iii.) generate output via the composition $\mathsf{dG} \leftarrow \Phi \leftarrow \mathsf{RG}$. Here Φ is responsible for "cleaning" the randomness produced by the possibly subverted randomness generator RG. In fact, we will see that this approach can fail even in the most generous setting when Φ is given by a random oracle and the adversary only subverts RG.

The attack is a straightforward adaptation of the techniques from [BPR14, BJK15]: the subverted implementation $\mathsf{RG}_{\text{IMPL}}$ can evaluate dG and appropriately query the random oracle Φ during the procedure of rejection sampling. It is easy, then, to arrange for the the output of $\mathsf{RG}_{\text{IMPL}}$ to be biased in a way only detectable by the adversary. While a generic attack is possible, for concreteness we present an attack on a subverted public-key cryptosystem which permits the adversary to effortlessly determine the (plaintext) message bit. This indicates that more advanced non-black-box techniques are necessary to remove steganographic channels in general.

Subverted randomness generation attack: In the following attack on pubic key encryption, the adversary honestly implements the key generation and decryption, and only subverts the encryption algorithm. Suppose the specification of the (public-key) encryption algorithm is defined as $\mathsf{Enc}_{\text{SPEC}} := (\mathsf{RG}_{\text{SPEC}}, \mathsf{dEnc}_{\text{SPEC}}, \Phi_{\text{SPEC}})$. The meaning of each component is self-evident: $\mathsf{RG}_{\text{SPEC}}$ generates uniformly random bits r_0, the function Φ_{SPEC} "cleans" r_0 to produces the final random bits r, and, finally, $\mathsf{dEnc}_{\text{SPEC}}$ takes the random bits r, the encryption key pk, and the message bit as inputs and produces a ciphertext C.

The attack: The adversary $\mathcal{A}$ first randomly chooses a backdoor z, and prepares the subverted implementation $\mathsf{Enc}^z_{\text{IMPL}} := (\mathsf{RG}_{\text{IMPL}}, \mathsf{dEnc}_{\text{IMPL}}, \Phi_{\text{IMPL}})$ with the backdoor embedded. In particular, $\mathsf{RG}_{\text{IMPL}} := \mathsf{RG}^z_{\text{IMPL}}$ carries out rejection sampling to ensure that the ciphertext encrypting 0 and the ciphertext encrypting 1 can be distinguished by applying a PRF (using z as the key); the algorithms $\mathsf{dEnc}_{\text{IMPL}}$ and Φ_{IMPL} are honestly implemented (that is, identical to the specifications). Later the adversary $\mathcal{A}$ can easily learn secret information (indeed, the plaintext) from the ciphertext generated by the subverted algorithms by applying the PRF (using her backdoor z as the key). See Figure 4 for detailed description.

Security analysis. Due to the rejection sampling condition, it is easy to see that the adversary defined in Figure 4 can determine the plaintext bit perfectly from the ciphertext. As for the *detection probability*, the randomness output by $\mathsf{RG}_{\text{SPEC}}$ is a uniform λ-bit string; in contrast, the randomness output by $\mathsf{RG}_{\text{IMPL}}$ is a string selected uniformly from a (random) subset S of $\{0,1\}^\lambda$ (determined by the PRF). The subset S consists of all strings that carry 0 and 1 to ciphertexts satisfying a criterion given by the PRF. Let us think of the PRF as a random function, that means the

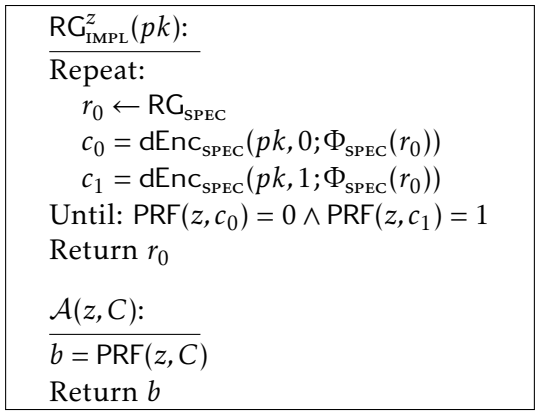


Figure 4: Subverted randomness generation and the message recovery algorithms

rejection sampling condition will be satisfied with probability 1/4 for each r_0 uniformly sampled. Essentially, we can consider S as a random subset of $\{0,1\}^\lambda$ with (expected) size $2^{\ell-2}$. If there is no collision when $\mathcal{W}$ asks q queries, then the q different bit strings observed by $\mathcal{W}$ can come from either of the whole space $\{0,1\}^\lambda$ or a subset S (with size larger than q). This means conditioned on no collision, no watchdog can tell apart $\mathsf{RG}_{\mathrm{IMPL}}$ from $\mathsf{RG}_{\mathrm{SPEC}}$. Using the total probability theorem, we can bound the distinguishing advantage by the probability that a collision appears in q queries when sample from S. We leave the full proof to Sec. D in the appendix.

Proposition 3.1. *For any CPA-secure (public-key) encryption scheme, for any public function Φ, the adversary $\mathcal{A}$ shown in Figure 4 can learn the plaintext with probability 1 given the ciphertext generated by $\mathsf{Enc}^z_{\mathrm{IMPL}}$ even if the randomness generation is separated and immunized by a random oracle. Furthermore, suppose $\mathsf{RG}_{\mathrm{IMPL}}$ outputs ℓ bits of randomness; then the detection advantage is $q^2/2^{\ell-4} + \mathsf{negl}(\lambda)$ for all PPT watchdogs making q queries during the interrogation, assuming PRF is a secure pseudorandom function.*

Remark 3.1. *We remark that for this particular attack we assume that the implementation has access to a public key—this yields an intuitively natural attack against a encryption scheme which permits full recovery of the message. However, the basic structure of the attack can be adapted to randomized algorithms in generality.*

3.2 Purifying randomness via double splitting

The attacks given above (and by [BPR14, BJK15]) demonstrate a core obstacle to defending randomized algorithms against subversion: the random coins drawn by the subverted implementation can be biased—for example, by rejection sampling—even if they are then "immunized" by a random oracle. On the other hand, the split-program model intuitively offers the watchdog an opportunity for fine-grained testing: Is the situation really no better than the purely black-box model? Let us take a closer look from the security analysis point of view.

Intuitively, specifications of the form $(\mathsf{RG}_{\mathrm{SPEC}}, \mathsf{dG}_{\mathrm{SPEC}}, \Phi_{\mathrm{SPEC}})$ can provide security in a kleptographic setting if the immunization function Φ can suitably interfere with generation of biased output by the implementation of $\mathsf{RG}_{\mathrm{SPEC}}$. To simplify our presentation, we assume throughout that $\mathsf{RG}_{\mathrm{SPEC}}$ produces at least λ bits of randomness; this does not affect the generality of the results. (Our techniques can be adapted to a low-entropy setting with some changes to running time of the

watchdog.[5]) An important feature in this setting is that an offline watchdog $\mathcal{W}$ can at least guarantee that the output r_0 of $\mathsf{RG}_{\text{IMPL}}$ is *unpredictable* to the adversary $\mathcal{A}$. Otherwise, the distribution given by $\mathsf{RG}_{\text{IMPL}}$ would have significant (non-negligible) collision probability,[6] which can be easily tested by $\mathcal{W}$ who simply draws two samples and rejects if it observes a collision. (As with the other tests we discuss, the success of this test can be amplified by repetition.) On the other hand, the collision probability of $\mathsf{RG}_{\text{SPEC}}$ is negligible. This suggests the intuition that $\Phi_{\text{SPEC}}(r_0)$ appears to $\mathcal{A}$ to be a randomly drawn value if Φ_{SPEC} is a random oracle. Unfortunately, $\mathcal{A}$ also holds the backdoor z which may contain information about the final output $r = \Phi(r_0)$ generated by the sampling and "cleaning" process. In particular, as shown in the attack, the subverted implementation has full access to Φ_{SPEC} and may thus bias the output $r = \Phi(r_0)$ as a function of z, which can be noticed by $\mathcal{A}$.

To circumvent the above obstacle, we introduce a new technique that further splits randomness generation into *two* random algorithms, $\mathsf{RG}^0_{\text{SPEC}}$ and $\mathsf{RG}^1_{\text{SPEC}}$, and combines their outputs using an immunization function Φ_{SPEC}; we shall see that this mechanism can destroy subliminal channels. In general, in the trusted amalgamation model, (see Definition A.1 in appendix. A.1) the user runs $\mathsf{RG}^0_{\text{IMPL}}$ and $\mathsf{RG}^1_{\text{IMPL}}$ independently and passes the joint outputs to Φ_{IMPL}; the final output will have the form $r = \Phi_{\text{IMPL}}(r_0 \circ r_1)$ (where $\circ$ denotes concatenation). The main idea behind this strategy is that it frustrates attempts by $\mathsf{RG}^0_{\text{IMPL}}$ and $\mathsf{RG}^1_{\text{IMPL}}$ to launch sophisticated rejection-sampling because the final output is not fully determined by either output. (In particular, neither can evaluate $\Phi_{\text{IMPL}}(r_0 \circ r_1)$ during the generation of r_0 or r_1.) In this way, if Φ_{SPEC} is modeled as a random oracle, the final output $\Phi_{\text{SPEC}}(r_0 \circ r_1)$ will be uncorrelated with $\mathcal{A}$'s state (which includes both $\mathcal{A}$'s random oracle queries and z). Now we can safely claim that r looks uniform even to $\mathcal{A}$.[7]

We remark on a similarity between the setting above and the topic of randomness extractors [NZ96]. Recall that a extractor E is a deterministic function which takes as input a number of imperfect random sources $X_1, X_2, \ldots, X_k$ and produces a nearly uniform output $E(X_1, \ldots, X_k)$. It is a fact that extraction is not possible when $k = 1$—no fixed deterministic function E can produce clean randomness from a source with bounded min-entropy. On the other hand, it is possible when $k \geq 2$ [CG88, Bou05, CZ15].

With this observation, we demonstrate a qualitative advantage of the split-program methodology. We first describe a stego-free specification of randomness generation in Fig. 5, then proceed to give an immunization strategy for arbitrary randomized algorithms so long as they have a pubic input distribution (or a small input domain). We apply these tools in next section to construct an IND-CPA public key encryption scheme that retains security under subversion; note that this is impossible to achieve if randomness generation is used as a black-box. (even if it is separated as an individual component).

Theorem 3.2. *Consider a randomness generation algorithm* RG *with specification* $(\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi_{\text{SPEC}})$ *as described in Fig. 5:* $\mathsf{RG}^0_{\text{SPEC}}$ *and* $\mathsf{RG}^1_{\text{SPEC}}$ *output uniformly random strings* r_0 *and* r_1 *of length* λ*, respectively, and* Φ_{SPEC} *is a hash function which takes* $r_0 \circ r_1$ *as input. Then* $\mathsf{RG}_{\text{SPEC}}$ *is stego-free in the*

[5] Observe that if $\mathsf{RG}_{\text{SPEC}}$ produces only $O(\log n)$ random coins then an offline watchdog, by a suitable regimen of repeated sampling, can empirically approximate (with high probability) the distribution of $\mathsf{RG}_{\text{IMPL}}$ with high accuracy. This can be directly compared with $\mathsf{RG}_{\text{SPEC}}$ using distance in total variation. Note that such a watchdog requires a number of samples polynomial in the resulting error.

[6] Observe that if D is a probability distribution on a set X, the optimal strategy for predicting the result of drawing an element of X according to D is simply to guess $\max_x D(x)$. If this maximum probability is ϵ, then the collision probability of D, equal to $\sum_x D(x)^2$, is at least ϵ^2.

[7] We remark that Φ_{IMPL} can be subverted, but it is a deterministic function with a public input distribution, the inconsistency can be ensured to be at only a negligible fraction of places due to the watchdog, see Lemma 2.2.

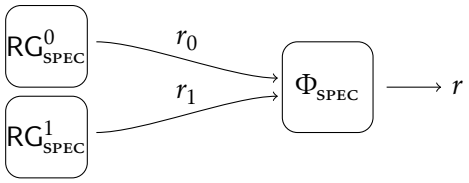


Figure 5: A stego-free specification for randomness generation.

trusted amalgamation model (see Def. A.1 in Sec. A.1 in the appendix) if Φ_{SPEC} *is modeled as a random oracle.*

Proof. If the specification of Fig. 5 is not stego-free, then for any offline watchdog $\mathcal{W}$ there is an adversary $\mathcal{A}$ that can prepare an implementation $\mathsf{RG}_{\text{IMPL}} := (\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}, \Phi_{\text{IMPL}})$ satisfying the following: (1.) $\mathcal{W}$ cannot distinguish $\mathsf{RG}_{\text{IMPL}}$ from $\mathsf{RG}_{\text{SPEC}}$ via oracle access; (2.) The adversary $\mathcal{A}$ can distinguish output of $\mathsf{RG}_{\text{IMPL}}$ from $\mathsf{RG}_{\text{SPEC}}$, i.e. she can win the game defined in Figure 12 in supporting material, Sec. A.1. We will then define an offline watchdog such that these two conditions cannot hold simultaneously for any adversary.

An offline watchdog. The watchdog $\mathcal{W}$'s strategy is as follows: $\mathcal{W}$ first draws a constant number of samples from $\mathsf{RG}^0_{\text{IMPL}}$ and $\mathsf{RG}^1_{\text{IMPL}}$; if $\mathcal{W}$ observes a collision in either distribution, it rejects the implementation outright (as collisions are negligible in SPEC). Next, $\mathcal{W}$ draws pairs of samples (again) from $\mathsf{RG}^0_{\text{IMPL}}$ and $\mathsf{RG}^1_{\text{IMPL}}$ and evaluates Φ_{IMPL} on (the concatenation of) each pair to ensure that the result is consistent with Φ_{SPEC}. (As usual, this testing involves only $O(1)$ samples and can be trivially amplified by repetition.)

Next, we will show, for any PPT adversary $\mathcal{A}$, if the detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is negligible, then the advantage $\mathbf{Adv}_{\mathcal{A}}$ will also be negligible, thus the two conditions cannot hold simultaneously.

Game transitions. We will go through the security game part of Def. A.1 step by step. W.l.o.g, we assume the challenge r contains only one element (i.e.,$q = 1$ in Def. A.1).

In Game-0, adversary $\mathcal{A}$ prepares subverted implementations $\mathsf{RG}_{\text{IMPL}} := (\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}, \Phi_{\text{IMPL}})$, and let Q be the set of random oracle queries $\mathcal{A}$ made during the manufacturing.

The challenger $\mathcal{C}$ samples from $\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}$ respectively and receives r_0, r_1, then $\mathcal{C}$ feeds $r_0 \circ r_1$ to Φ_{IMPL} and sends the output r as the challenge to $\mathcal{A}$. Let Q_b) (for $b = 0, 1$) be the set of random oracle queries made by $\mathsf{RG}^b_{\text{IMPL}}$ before outputting r_b; All the random oracle queries will be answered with randomly chosen values.

Game-1 is identical to Game-0, except that Φ_{IMPL} is replaced with Φ_{SPEC}; Game-2 is identical to Game-1, except that the challenger $\mathcal{C}$ simply chooses a uniform r and directly sends it to $\mathcal{A}$ as the challenge; Game-3 is identical to Game-2, except that $\mathsf{RG}_{\text{IMPL}}$ is completely replaced with $\mathsf{RG}_{\text{SPEC}}$; Game-4 is identical to Game-3, except that r is generated as in Game-0, but the challenger uses $\mathsf{RG}_{\text{SPEC}}$ instead.

Probabilistic analysis. We will analyze the gaps of each game transition conditioned on the event that $\mathbf{Det}_{\mathcal{W},\mathcal{A}}$ is negligible, we denote this event as $D_{\mathcal{W}}$. All the probability gap would be under the condition $D_{\mathcal{W}}$. For brevity, we assume the condition $D_{\mathcal{W}}$ without mention for the analysis of each probability gap.

First, since Φ_{SPEC} is a deterministic function with a public input distribution (the output

distribution of $\mathsf{RG}^0_{\text{IMPL}} \times \mathsf{RG}^1_{\text{IMPL}}$), following Lemma 2.2,

$$\Pr\left[\Phi_{\text{IMPL}}(r_0 \circ r_1) \neq \Phi_{\text{SPEC}}(r_0 \circ r_1) : r_0 \leftarrow \mathsf{RG}^0_{\text{IMPL}}, r_1 \leftarrow \mathsf{RG}^1_{\text{IMPL}}\right] \leq \mathsf{negl}(\lambda).$$

Otherwise, the gap is non-negligible and the watchdog $\mathcal{W}$ will notice the inconsistency (with non-negligible probability). It follows that replacing Φ_{IMPL} with Φ_{SPEC} would incur only a negligible difference (conditioned on $D_{\mathcal{W}}$), thus:

$$|\Pr[b_{\mathcal{C}} = 1 \text{ in Game-0}] - \Pr[b_{\mathcal{C}} = 1 \text{ in Game-1}]| \leq \mathsf{negl}(\lambda).$$

Second, we will argue that the probability that $r_0 \circ r_1$ is ever queried (falling in $Q \cup Q_0 \cup Q_1$) is negligible, and now we are in Game-1 using Φ_{SPEC} which is a random oracle.

It is easy to see that $\Pr[r_0 \circ r_1 \in Q] \leq \mathsf{negl}(\lambda)$; otherwise, the watchdog will observe a collision in $\mathsf{RG}^i_{\text{IMPL}}$ with non-negligible probability. To see this, let $R_0 = \{r_0 \mid \exists r_1, r_0 \circ r_1 \in Q\}$, note that Q, R_0 are only polynomially large. If the probability $r_0 \circ r_1$ falls into Q (thus the probability that r_0 falls into R_0) is non-negligible, say δ, that means r_0 will be generated by $\mathsf{RG}^0_{\text{IMPL}}$ with probability at least $\delta_0 = \delta/\text{poly}(\lambda)$. It follows that the collision probability that $\mathsf{RG}^0_{\text{IMPL}}$ produces the same output r_0 would be δ_0^2. While on the other hand, $\mathsf{RG}^0_{\text{SPEC}}$ produces uniform bits, the collision probability (that $\mathsf{RG}^0_{\text{IMPL}}$ produces the same uniform output string r_0) would be negligible. Thus the watchdog can easily distinguish $\mathsf{RG}^0_{\text{IMPL}}$ from $\mathsf{RG}^0_{\text{SPEC}}$ when drawing, say 2 samples.

Similarly, we bound the probability for Q_0, Q_1. Let $R_{0,1} = \{r_1 \mid \exists r_0, r_0 \circ r_1 \in Q_0\}$. Since $\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}$ are independently run, the probability that r_1 falls into the polynomially large set $R_{0,1}$ would be negligible; otherwise, $\mathsf{RG}^1_{\text{IMPL}}$ outputs r_1 with a non-negiligble probability, then $\mathcal{W}$ can notice the difference between implementations and the specifications by identifying collisions. Thus $\Pr[r_0 \circ r_1 \in Q_0] \leq \Pr[r_1 \in R_{0,1}] \leq \mathsf{negl}(\lambda)$. The same holds for Q_1.

The adversary $\mathcal{A}$ is holding the set of random oracle queries Q, and a backdoor z. The only way r may correlate with z is that $r_0 \circ r_1$ is queried during the execution of $\mathsf{RG}^0_{\text{IMPL}}$, or $\mathsf{RG}^1_{\text{IMPL}}$. If $r_0 \circ r_1 \notin Q \cup Q_0 \cup Q_1$, $r_0 \circ r_1$ will be independent with $\mathcal{A}$'s view (Q, z), thus $r = \Phi_{\text{SPEC}}(r_0 \circ r_1) = \mathsf{RO}(r_0 \circ r_1)$ looks uniform to $\mathcal{A}$. We can claim that:

$$|\Pr[b_{\mathcal{C}} = 1 \text{ in Game-1}] - \Pr[b_{\mathcal{C}} = 1 \text{ in Game-2}]| \leq \mathsf{negl}(\lambda).$$

Next, it is easy to see that $\Pr[b_{\mathcal{C}} = 1 \text{ in Game-2}] = \Pr[b_{\mathcal{C}} = 1 \text{ in Game-3}]$ since the adversary receives the identical challenge. Also, $\Pr[b_{\mathcal{C}} = 1 \text{ in Game-3}] = \Pr[b_{\mathcal{C}} = 1 \text{ in Game-4}]$ since querying $\mathsf{RG}_{\text{SPEC}}$ yields a uniform output $\mathsf{RO}(u_0 \circ u_1)$, where u_0, u_1 are uniformly chosen.

To conclude, conditioned on $D_{\mathcal{W}}$, we have:

$$|\Pr[b_{\mathcal{C}} = 1 \text{ in Game-0}] - \Pr[b_{\mathcal{C}} = 1 \text{ in Game-4}]| \leq \mathsf{negl}(\lambda).$$

Observe that Game-0 corresponds to the case that challenger flips a coin to be 0, i.e., $\mathcal{C}$ uses $\mathsf{RG}_{\text{IMPL}}$ to generate the challenge messages, while Game-4 corresponds to the case that $b = 1$, when $\mathcal{C}$ uses $\mathsf{RG}_{\text{SPEC}}$. It follows that:

$$\mathbf{Adv}_{\mathcal{A}} = |\Pr[b_{\mathcal{C}} = 1] - 1/2| \leq \mathsf{negl}(\lambda).$$

Combine all above, we can conclude that the $\mathsf{RG}_{\text{SPEC}}$ defined in Figure 5 is stego-free. □

Implementation considerations. Practical deployment of such splitting and amalgamation—especially as it requires independence and (for the watchdog) copying internal state—clearly requires detailed consideration of the particular computational environment. In general, there are two natural approaches to achieve independence: The most modular approach relies on modern lightweight virtualization layers such as Docker [Doc] to insulate individual copies of the adversary's code; we remark that this also permits state duplication (which may be necessary for the watchdog in the stateful case). More aggressive complete virtualization is also possible, but more cumbersome. An alternate approach relies on constraining the source code (or the compiler) to directly limit I/O and system calls; this has the advantage that the components can be run efficiently in the native environment. Finally, there may also be settings where it is possible to isolate the program in the architectural/hardware layer or physically separate various components (e.g., using Intel's secure isolation gateway, or even move one RG^i outside the user's computer, use a random beacon, etc.).

Remark 3.3 (Stateful algorithms). *The previous construction gives a concrete analysis of how a simple watchdog can secure a split specification. At the end of Section 2, we remarked that our results hold even if the implementations are stateful, in the sense that they may maintain local state that persists between executions.*

To see this in the example above, we need to ensure that the implementation $\mathsf{RG}^b_{\text{IMPL}}$ produces unpredictable outputs (even to the adversary) for polynomially many invocations. Even if the implementation keeps some internal state, we can recover the same result by adopting a slightly more advanced watchdog which is permitted to have running time that depends on the adversary. In particular, if the adversary is permitted to run the implementation k times, the watchdog may, for each $1 \leq k' \leq k$, (1.) run the implementation k' times to generate a particular internal state s, and then (2.) run the implementation a constant number of times with state s to test for collisions. By this process, the watchdog can guarantee the entropy of a sample conditioned on previous samples. A straightforward analysis shows that if there is a particular k' for which the resulting distribution has non-negligible collision probability (with non-negligible probability in the state produced by the first $k'-1$ invocations), the watchdog can detect this (with non-negligible probability). This was also observed in [RTYZ15]. Note that the actual procedure carried out by the watchdog is still universal, but the running time (and number of samples) may be adapted to the adversary. Note also that no strictly universal watchdog (whose running time is independent of the adversary) can possibly detect an algorithm that suddenly becomes deterministic after some (polynomial) number of steps determined by the adversary.

We further remark that the above can be easily generalized to stateful algorithm having public input distributions.

3.3 Stego-free specifications for randomized algorithms

Now we are ready to establish the general result which yields a stego-free specification for any randomized algorithm in the trusted amalgamation model (see Definition A.2 in Sec. A.2 in the appendix). We use a randomized algorithm with a public input distribution as a running example; this can be generalized directly to the setting with an extra small input domain. As discussed in Section 2, this already covers many of the interesting cases such as key generation and bit encryption.

With a stego-free specification of randomness generation $\mathsf{RG}_{\text{SPEC}}$ (with trusted amalgamation) the simple split-program specification (where a randomized algorithm that is split into $\mathsf{RG}_{\text{SPEC}}$ and

the deterministic $\mathsf{dG}_{\mathrm{SPEC}}$) is a stego-free specification for the functionality G. In particular, $\mathsf{dG}_{\mathrm{SPEC}}$ takes r, x as inputs where r is the final output of $\mathsf{RG}_{\mathrm{SPEC}}$, and x is generated by an instance generator $\mathsf{IG}_{\mathrm{IMPL}}$ that can be implemented by the adversary; see Fig. 6.

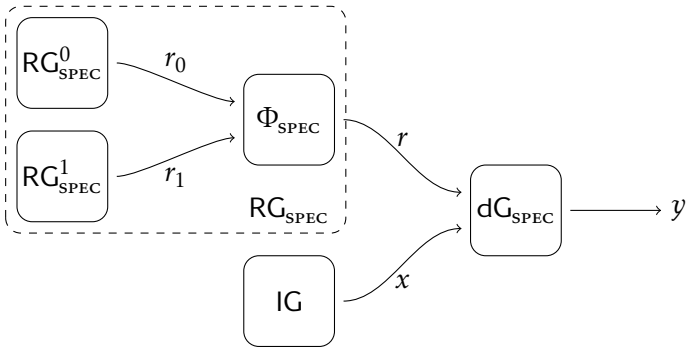


Figure 6: A stego-free specification for randomized algorithm G, where x is obtained from IG.

Security analysis. The intuition is as follows: We know that a subverted implementation $\mathsf{dG}_{\mathrm{IMPL}}$ of a deterministic algorithm will be consistent with its specification with an overwhelming probability when the inputs are sampled from a public distribution (Lemma 2.2). In this case, it is generated by $\mathsf{IG}_{\mathrm{IMPL}} \times \mathsf{RG}_{\mathrm{IMPL}}$. Thus $\mathsf{dG}_{\mathrm{IMPL}}$ can be considered as good as $\mathsf{dG}_{\mathrm{SPEC}}$ when restricted to this public input distribution. Furthermore, $\mathsf{RG}_{\mathrm{SPEC}}$ is stego-free; the implementation can be used as the specification which outputs a uniform string as well! Then all implementations have been replaced with their specifications, and the security follows easily. Note that the $\mathsf{IG}_{\mathrm{IMPL}}$ in Figure 6 is not included in $\mathsf{G}_{\mathrm{SPEC}}$. Formally we present the following theorem, and we defer the detailed proof to appendix. D.

Theorem 3.4. *For any randomized algorithm* G, *consider the specification* $\mathsf{G}_{\mathrm{SPEC}} := (\mathsf{RG}_{\mathrm{SPEC}}, \mathsf{dG}_{\mathrm{SPEC}})$, *where* $\mathsf{RG}_{\mathrm{SPEC}}$ *generates* λ *bits of uniform randomness and* $\mathsf{dG}_{\mathrm{SPEC}}$ *is deterministic. Let* $(\mathsf{RG}^0_{\mathrm{SPEC}}, \mathsf{RG}^1_{\mathrm{SPEC}}, \Phi_{\mathrm{SPEC}})$ *be the double-split specification of* $\mathsf{RG}_{\mathrm{SPEC}}$ *as above. If (1)* $\mathsf{RG}^0_{\mathrm{SPEC}}$ *and* $\mathsf{RG}^1_{\mathrm{SPEC}}$ *output* λ *uniform bits; (2)* Φ_{SPEC} *takes* $r_0 \circ r_1$ *as input, and outputs* r*; (3)* $\mathsf{dG}_{\mathrm{SPEC}}$ *is deterministic and it takes* r *and* x *as input, where* x *is generated by a sampler* $\mathsf{IG}_{\mathrm{IMPL}}$ *(see Fig. 6), then* $\mathsf{G}_{\mathrm{SPEC}}$ *is stego-free with a trusted amalgamation (according to Def. A.2 in appendix. A.2). Here* Φ_{SPEC} *is modeled as a random oracle, and* $\Phi^0_{\mathrm{IMPL}}, \Phi^1_{\mathrm{IMPL}}$ *are executed independently.*

It is straightforward to generalize the result above to support algorithms with an extra small size input as they essentially expands the input distribution to be q distributions, and the watchdog can check all of them.

Corollary 3.5. *For any randomized algorithm* G, *consider a specification* $\mathsf{G}_{\mathrm{SPEC}} := (\mathsf{RG}_{\mathrm{SPEC}}, \mathsf{dG}_{\mathrm{SPEC}})$, *where* $\mathsf{RG}_{\mathrm{SPEC}}$ *is defined as* $(\mathsf{RG}^0_{\mathrm{SPEC}}, \mathsf{RG}^1_{\mathrm{SPEC}}, \Phi_{\mathrm{SPEC}})$ *(as in Fig. 5): If* $\mathsf{dG}_{\mathrm{SPEC}}$ *takes* r, x *and* m *as input, where* x *is generated by a sampler* $\mathsf{IG}_{\mathrm{IMPL}}$, *and* m *is taken from a polynomial size public domain* D. *Then the specification* $\mathsf{G}_{\mathrm{SPEC}}$ *is stego-free with a trusted amalgamation (according to Def. A.2 in appendix Sec. A.2) if* Φ_{SPEC} *is modeled as a random oracle.*

4 Subversion-Resistant Encryption

Encryption is a fundamental cryptographic tool, and is well-understood in conventional settings. Unfortunately, encryption in the kleptographic setting has been a particular challenge; existing

solutions do not offer satisfactory CPA security or protection from subversion of key generation. The main motivation for this paper is to develop subversion resistant encryption schemes (both symmetric and asymmetric) that meet these elementary criteria.

To briefly survey the state-of-the-art: Previous results [BPR14, DFP15, BH15], in order to protect against subliminal channel attacks, adopt a special kind of *deterministic* encryption scheme. Note, however, that IND-CPA security is impossible for deterministic public-key encryption (or deterministic, stateless symmetric-key encryption). (An important exception is [DMSD15], which assumes that a trusted, subversion-free, party re-randomizes all communication.) These results permit only the encryption algorithm to be subverted: key generation and decryption are assumed to be honest. A further "decryptability" assumption was adopted in [BPR14, BH15] to ensure security; it assumes that every ciphertext generated by the subverted implementation can be correctly decrypted using the specification. While this helps to achieve security, it seems difficult to justify; see the criticism in [DFP15]. In general, then, correctness is not placed on the same footing as security (which is established via the specification in tandem with watchdog testing), but is rather provided by fiat.

In this section, we will address all these concerns.

4.1 Subversion-resistant symmetric encryption

We first construct subversion-resistant symmetric-key encryption in the case when all algorithms are subject to subversion. We then discuss correctness and how to remove the "decryptability" assumption.

Defining subversion resistance and correctness. We follow the definitional framework of [RTYZ15] to define a subversion-resistant symmetric-key encryption scheme. The definition formalizes the intuition that an encryption scheme is secure under subversion if IND-CPA security can be preserved in the "subverted security game" conditioned on the implementation evading detection. Recall that in the "subverted security game" the challenger uses the subverted implementations of all the algorithms. Initially, we focus on *stateless* encryption algorithms, and discuss how to relax this assumption when we discuss large message spaces.

Definition 4.1. *A (stateless) symmetric-key (bit) encryption scheme with specification* $\mathbf{E}_{\text{SPEC}} := (\mathsf{KG}_{\text{SPEC}}, \mathsf{Enc}_{\text{SPEC}}, \mathsf{Dec}_{\text{SPEC}})$ *is subversion-resistant in the offline watchdog model if there exists a* PPT *watchdog* $\mathcal{W}$ *so that, for any* PPT *adversary* $\mathcal{A}$ *playing the game described in Figure 7, either*

$$\mathbf{Adv}_{\mathcal{A}} \text{ is negligible,} \quad \text{or} \quad \mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible.}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr[b_{\mathcal{C}} = 1] - 1/2|$, *and,* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathbf{E}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\Pr[\mathcal{W}^{\mathbf{E}_{\text{SPEC}}}(1^\lambda) = 1]\right|$.

Correctness under subversion is an overlooked, but fundamental property. For example, one can imagine that the adversary "hates" a certain message m that is unknown to the watchdog (e.g., "your cryptosystem is subverted"). The subverted implementation can then check whether the plaintext matches m and, if so, $\mathsf{Dec}_{\text{IMPL}}$ outputs an arbitrary value other than m. This can be used by the adversary to effectively implement censorship. We say a symmetric-key encryption scheme is *correct* under subversion if the following holds:

$$\forall M, \Pr\left[\begin{array}{l} \mathsf{Dec}_{\text{IMPL}}(C) \neq M : \\ C \leftarrow \mathsf{Enc}_{\text{IMPL}}(K, M), K \leftarrow \mathsf{KG}_{\text{IMPL}}(1^\lambda) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

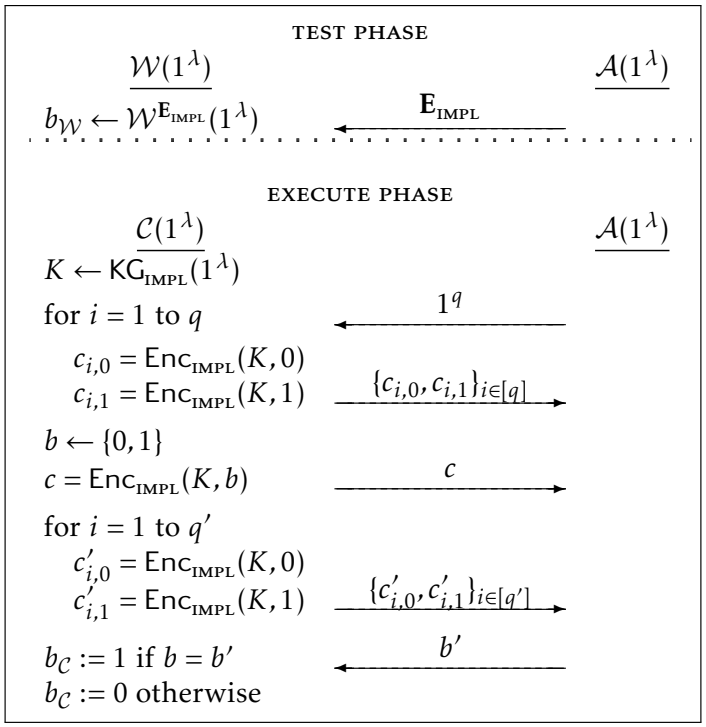


Figure 7: Subversion-resistant symmetric-key bit encryption. (The stateless case.)

where the probability is over the choice of K and the coins used in $\mathsf{Enc}_{\text{IMPL}}$.

Constructing subversion resistant symmetric-key bit encryption. We proceed to design a specification for any secure symmetric-key encryption scheme so that the subliminal channels in all of the algorithms can be destroyed. With the general tool we developed in Section 3.2, we will "immunize" the algorithms one by one.

First, the (symmetric-key) key generation algorithm simply takes the security parameter as input and produces a random element in the key space. Following Thm. 3.2 directly, the specification $\mathsf{KG}_{\text{SPEC}} := (\mathsf{KG}^0_{\text{SPEC}}, \mathsf{KG}^1_{\text{SPEC}}, \Phi^{\mathsf{KG}}_{\text{SPEC}})$ (see Fig. 8) is stego-free; here $\mathsf{KG}^0_{\text{SPEC}}, \mathsf{KG}^1_{\text{SPEC}}$ both output random elements in the key space $\mathcal{K}$, and $\Phi^{\mathsf{KG}}_{\text{SPEC}} : \mathcal{K} \times \mathcal{K} \to \mathcal{K}$, is modeled as a random oracle. As discussed above, the random oracle assumption can be removed if we allow randomness generation to be further subdivided; see appendix. B.

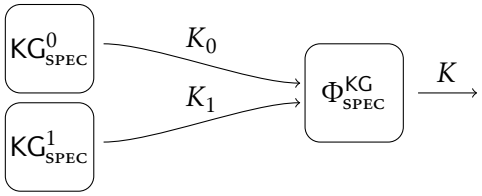


Figure 8: Stego-free specification for key generation algorithm in a symmetric-key bit encryption scheme.

Next, consider the encryption algorithm; we focus only on bit encryption. In this case, the

encryption algorithm takes three inputs: a security parameter, a small domain input (a bit), and a pair given by the random coin and the key, which come from a public input distribution $\mathsf{RG}_{\text{IMPL}} \times \mathsf{KG}_{\text{IMPL}}$. From Corollary 3.5, the specification $\mathsf{Enc}_{\text{SPEC}} := (\mathsf{RG}_{\text{SPEC}}, \mathsf{dEnc}_{\text{SPEC}})$ (as described in Fig. 9), where $\mathsf{RG}_{\text{SPEC}} := (\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi^{\mathsf{RG}}_{\text{SPEC}})$ defined as in Fig. 5, is stego-free if Φ_{SPEC} is assumed to be a random oracle.

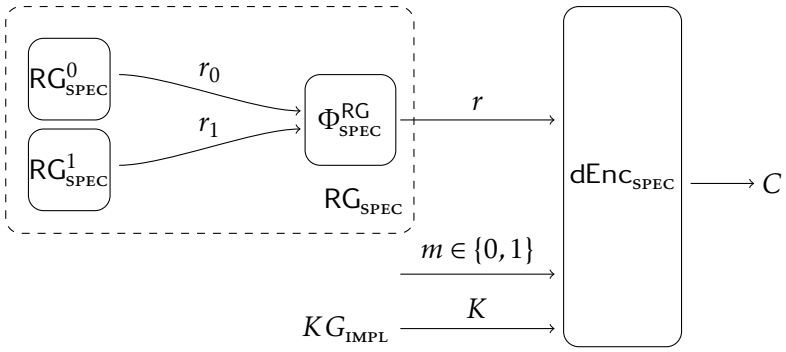


Figure 9: Stego-free specification for encryption algorithm in a symmetric-key bit encryption scheme.

Finally, consider the decryption algorithm. This algorithm does not influence CPA security, but directly influences correctness. Observe that the decryption algorithm is deterministic and can be analyzed with a public input distribution as we focus on bit encryption. To see this, the decryption algorithm will input a key K (generated by $\mathsf{KG}_{\text{IMPL}}$) and a ciphertext C (generated by encrypting 0 or 1 using K). The watchdog can sample from the input distribution of $\mathsf{Dec}_{\text{IMPL}}$ to check the correctness, i.e., the consistency with $\mathsf{Dec}_{\text{SPEC}}$.

With all of the algorithms handled individually as above, we present the first general immunizing result for randomized encryption algorithms.

Theorem 4.2. *Given any stateless IND-CPA secure symmetric bit encryption scheme, the specification described above is subversion resistant and correct according to Def. 4.1.*

(sketch). The specification is described above (with the randomized algorithm split into randomness generation and a deterministic part, and the randomness generation split into two components together with an immunizing function as in Figure 5.).

Security. The watchdog is the combination of those described above: it guarantees that $\mathsf{RG}_{\text{SPEC}}$ is stego-free (making samples to observe collisions as in the proof of Theorem 3.2) and guarantees that $\mathsf{dEnc}_{\text{IMPL}}$ is consistent with the specification on inputs sampled from $\mathsf{KG}_{\text{IMPL}} \times \mathsf{RG}_{\text{IMPL}}$ (with 0 and 1); cf. Lemma 2.2. Here we only sketch the game changes and explain the negligible differences arising during the game transitions conditioned the watchdog's result.

Game-0 is the original game as described in Figure 7 with a trusted amalgamation; Game-1 is the same as Game-0 except $\mathsf{KG}_{\text{IMPL}}$ is replaced with $\mathsf{KG}_{\text{SPEC}}$; Game-2 is the same as Game-1 except $\mathsf{Enc}_{\text{IMPL}}$ is replaced with $\mathsf{Enc}_{\text{SPEC}}$.

The adversary's advantage in Game-0 and Game-1 are negligibly close because of the stego-freeness of $\mathsf{KG}_{\text{SPEC}}$. To see this, a simulator can simulate the rest of the games in Game-0 (or Game-1) after receiving K sent by the challenger in the game defining stego-freeness for $\mathsf{KG}_{\text{SPEC}}$. If one can distinguish Game-0 from Game-1, then the simulator can easily tell apart $\mathsf{KG}_{\text{IMPL}}$ from $\mathsf{KG}_{\text{SPEC}}$

(even if the challenger does the amalgamation). Similarly, we can argue Game-1 and Game-2 are indistinguishable because $\mathsf{Enc}_{\text{SPEC}}$ is stego-free. The fact that $\mathsf{Enc}_{\text{SPEC}}$ is stego-free follows from Corollary 3.5.

Of course, in Game-2 every algorithm used in the game is faithfully implemented; thus the IND-CPA security of the underlying encryption scheme would hold.

Regarding correctness. As described above, the decryption algorithm for a bit encryption scheme has a public input distribution. The watchdog can sample random keys and check whether $\mathsf{Dec}_{\text{IMPL}}$ works properly for bit encryption. Following Lemma 2.2, we conclude that: $\forall b \in \{0,1\}$,

$$\Pr\left[\mathsf{Dec}_{\text{IMPL}}(K, \mathsf{Enc}_{\text{SPEC}}(K,b)) \neq b : K \leftarrow \mathsf{KG}_{\text{IMPL}}\right] \leq \mathsf{negl}(\lambda),$$

while for the encryption algorithm, $\mathsf{Enc}_{\text{IMPL}}$ can be used interchangeably with $\mathsf{Enc}_{\text{SPEC}}$ assuming trusted amalgamation. Thus, $\forall b \in \{0,1\}$,

$$\Pr[\mathsf{Dec}_{\text{IMPL}}(\mathsf{Enc}_{\text{IMPL}}(K,b)) \neq b : K \leftarrow \mathsf{KG}_{\text{IMPL}}] \leq \mathsf{negl}(\lambda).$$

Combining these yields the statement of the theorem. □

Subversion resistant symmetric encryption with large message spaces. (Also handling state). For large message spaces, the security game must be adapted to allow the adversary to query; see Figure 14 in appendix. A. As mentioned in Section 2, this immediately invites an input-trigger attack. Specifically, for a particular query m_i (chosen randomly by A during subversion), the subverted encryption algorithm may directly output the secret key; the same threat exists for stateful *bit* encryption, where a particular sequence of encryption bits may act as a trigger. Note that this simply cannot be detected by a PPT watchdog (making polynomially many samples). Furthermore, the same attack can be launched on $\mathsf{Dec}_{\text{IMPL}}$ to ruin correctness (as $\mathsf{Dec}_{\text{IMPL}}$ can output a different value). This suggests the principle that a subverted implementation should never be given a "naked" input generated by the adversary, e.g., the queried message. (Such an input can act as a direct trigger when drawn from a large space, or can be remembered by stateful subversions to effectively simulate a large space.)

However, observe that if the input message can be forced to come from a known distribution U (e.g., the uniform distribution) the watchdog can check consistency between $\mathsf{dEnc}_{\text{IMPL}}$ and $\mathsf{dEnc}_{\text{SPEC}}$ on U (we ignore the other inputs here for simplicity, since they are either fixed or from a public input distribution). Indeed, the watchdog can guarantee that with overwhelming probability $\mathsf{dEnc}_{\text{IMPL}}$ is as good as $\mathsf{dEnc}_{\text{SPEC}}$ when the input is sampled according to U. In this case, the watchdog can in fact test the behavior of the algoriothm on a *sequence* of such inputs from U, which is how we handle stateful subversion. Now, to defend against an input-trigger attack, we must ensure that the probability that any *particular* m causes an inconsistency is negligible. This sort of "worst-case vs. average-case" relationship arises in the theory of *self-correcting programs* [Rub91], where one wants to transform a program with a negligible fraction of bad inputs into a negligible error probability for every input. With these observations, we return to the large-message space challenge.

Constructions. First, we consider straightforward bit-wise encryption. When encrypting a message $m = m_1 \ldots m_\ell$ for $\ell = |m|$, the user generates the ciphertext by calling $\mathsf{Enc}_{\text{IMPL}}$ ℓ times, yielding $C := (c_1, \ldots, c_\ell)$, where $c_i = \mathsf{Enc}_{\text{IMPL}}(K, m_i)$. Since the bit encryption scheme we developed above preserves IND-CPA security; IND-CPA security follows via a simple hybrid argument. (Note, however, that in this case it is important that the encryption algorithm is stateless.)

To develop a more efficient scheme that can further handle long message (also stateful encryption) we augment the model by permitting the user (i.e., the challenger) to carry out *one trusted addition* operation for each encryption/decryption, see Fig. 10. (We continue to assume trusted amalgamation, as before.) We augment the specification of the encryption algorithm with a random message generator $\mathsf{MG}_{\text{SPEC}}$. Specifically, the specification of the encryption algorithm $\mathsf{Enc}_{\text{SPEC}}$ has the form $(\mathsf{RG}_{\text{SPEC}}, \mathsf{dEnc}_{\text{SPEC}}, \mathsf{MG}_{\text{SPEC}})$, where $\mathsf{MG}_{\text{SPEC}}$ has the specification $(\mathsf{MG}^0_{\text{SPEC}}, \mathsf{MG}^1_{\text{SPEC}}, \Phi^{\mathsf{MG}}_{\text{SPEC}})$, and $\mathsf{RG}_{\text{SPEC}}$ is as before (as in Fig. 5). When encrypting a message m, the user first runs $\mathsf{MG}_{\text{IMPL}}$ (the implementation) to sample a random message u, and computes $m' = m \oplus u$. The user will call $\mathsf{Enc}_{\text{IMPL}}$ to encrypt the new message m' and obtains the ciphertext c'. This includes calls to $\mathsf{KG}_{\text{IMPL}}, \mathsf{RG}_{\text{IMPL}}$ and passing the corresponding outputs K, r together with m' to $\mathsf{dEnc}_{\text{IMPL}}$; see Figure 10. Observe that m' is a *uniformly chosen message* (as the watchdog can ensure that u is safely generated). The new ciphertext C now includes u together with the ciphertext c'. For decryption, the user first runs $\mathsf{Dec}_{\text{IMPL}}$ on c' and obtains m'; then the user computes $m = m' \oplus u$.

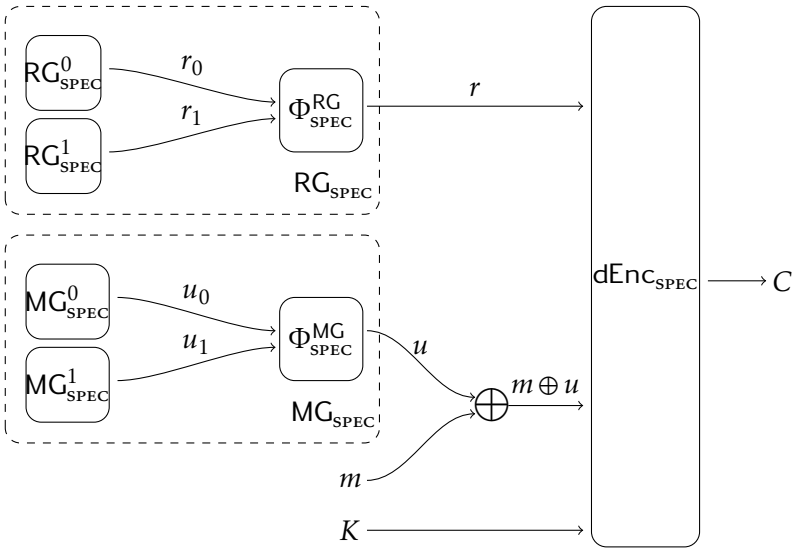


Figure 10: Stego-free encryption specification supporting large messages, where $K \leftarrow \mathsf{KG}_{\text{IMPL}}$.

Security analysis. The intuition that this simple method works is as follows: First, we generalize Theorem 4.2 that symmetric-key encryption for *random messages* are also subversion resistant. To see this, we analyze the stego-freeness algorithm by algorithm. The key generation is the same as the bit encryption. The encryption algorithm now takes input a uniform message, together with the key and security parameter. It means that the encryption algorithm (the deterministic part $\mathsf{dEnc}_{\text{SPEC}}$) now takes inputs from public input distributions, i.e., $\mathsf{KG}_{\text{IMPL}} \times \mathsf{RG}_{\text{IMPL}} \times \mathcal{U}_{\mathcal{M}}$, where $\mathcal{U}_{\mathcal{M}}$ is the uniform distribution over message space $\mathcal{M}$. Following Theorem 3.4, such encryption algorithm will be stego-free as long as the specification is designed as Figure 6. Now for the decryption algorithm, since the encryption is for uniform messages, thus the decryption algorithm now also takes a public input distribution. Next, we show the encryption specification defined in Figure 10 indeed takes uniform messages as input. (1.) Following Theorem 3.2, the uniform message sampler $\mathsf{MG}_{\text{SPEC}}$ is stego-free. (2.) With the trusted addition operation, when a that $\mathsf{Enc}_{\text{IMPL}}$ takes as input will be $m' = m \oplus u$, where u looks uniform even to the adversary, thus m' would look uniform to $\mathsf{Enc}_{\text{IMPL}}$ (actually the deterministic part $\mathsf{dEnc}_{\text{IMPL}}$). Similar to the analysis of Theorem 4.2, we can

show a stronger result that handles the correctness and subversion resistance for symmetric-key encryption supporting long messages. The full stateful case was handled in detail in remark 3.3.

Theorem 4.3. *For any IND-CPA secure symmetric-key encryption, the specification described as above is subversion resistant and correct according to Def. A.3, assuming a trusted $\oplus$ operation and amalgamation.*

4.2 Public key encryption preserving IND-CPA security under subversion

Now we turn to public-key encryption, which follows fairly directly from the previous construction. The major difference arises with key generation, as asymmetric key generation has to be treated with more care than simple randomness generation; see Figure 11, which indicates the construction. Specifically, the basic techniques used for symmetric key encryption above can be adapted for public key encryption. Key generation must be considered as a randomized algorithm producing structured output (with only security parameter as input). With these tools at hand, we resolve the major open problem to construct a IND-CPA secure PKE when facing subversions that we asked at the beginning of the paper. We remark that the assumption of "trusted $\oplus$" in the theorem below can be removed if the message space is small.

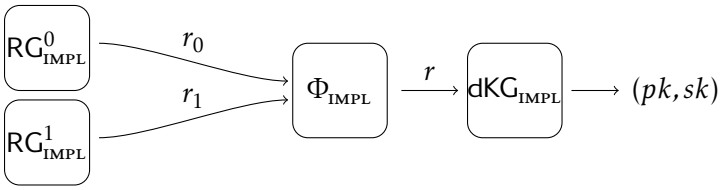


Figure 11: Stego-free asymmetric key generation.

Corollary 4.4. *For any IND-CPA secure public key bit encryption, there exists a specification design such that it is subversion resistant and correct with trust amalgamation. Furthermore, for any IND-CPA public key encryption (supporting large input space), there exists a specification design such that it is subversion resistant and correct according to Def. A.4, if the user can do a trusted $\oplus$ and amalgamation.*

References

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 364–375. ACM Press, October 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 Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 1431–1440. ACM Press, October 2015.

[Bou05] Jean Bourgain. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory*, 1:1–32, 2005.

[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.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Computing.*, 17(2):230–261, 1988.

[CZ15] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functionss. Technical report, TR15-119, Electronic Colloquium on Computational Complexity, 2015.

[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.

[Doc] Docker.Inc. https://www.docker.com/.

[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.

[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.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. http://eprint.iacr.org/2015/1162.

[RTYZ15] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. Cryptology ePrint Archive, Report 2015/695, 2015. http://eprint.iacr.org/2015/695.

[Rub91] Ronitt A. Rubinfeld. *A Mathematical Theory of Self-checking, Self-testing and Self-correcting Programs*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1991. UMI Order No. GAX91-26752.

[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

A.1 Stego-free randomness generation.

One of the most fundamental components of a randomized algorithm is the randomness generation. In particular, the devastating attacks using subliminal channels are exploring the subverted randomness generation. Moreover, if we have clean randomness, many tasks (including the CPA-secure public key encryption) which are previously considered impossible may become reachable. Formally,

Definition A.1. *For any randomness generation algorithm* RG*, consider a specification* $\mathsf{RG}_{\text{SPEC}} := (\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi_{\text{SPEC}})$*. We say such specification* $\mathsf{RG}_{\text{SPEC}}$ *is* stego-free with a trusted amalgamation in the offline watchdog model, *if there exists an offline* PPT *watchdog* $\mathcal{W}$*, for any* PPT *adversary* $\mathcal{A}$ *playing the following game (see Figure 12) with challenger* $\mathcal{C}$*, such that, at least one of the following conditions hold:*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible, or, } \mathbf{Adv}_{\mathcal{A}} \text{ is negligible,}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \left|\Pr[b_{\mathcal{C}} = 1] - \frac{1}{2}\right|$*, and* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{RG}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathsf{RG}_{\text{SPEC}}}(1^\lambda) = 1]\right|$*,*

A.2 Stego-free randomized algorithms with public input distributions

Definition A.2. *For a randomize algorithm* G*, consider a specification* $\mathsf{G}_{\text{SPEC}} := (\mathsf{RG}_{\text{SPEC}}, \mathsf{dG}_{\text{SPEC}})$*, where* $\mathsf{RG}_{\text{SPEC}} := (\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi_{\text{SPEC}})$*, and* $\mathsf{dG}_{\text{SPEC}}$ *is deterministic. We say such specification* G_{SPEC} *is* stego-free with a trusted amalgamation in the offline watchdog model, *if there exists an offline* PPT *watchdog* $\mathcal{W}$*, for any* PPT *adversary* $\mathcal{A}$ *playing the following game (see Figure 13) with challenger* $\mathcal{C}$*, such that, either,*

$$\mathbf{Adv}_{\mathcal{A}} \text{ is negligible, or, } \mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible.}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \left|\Pr[b_{\mathcal{C}} = 1] - \frac{1}{2}\right|$*, and,* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{W}^{\mathsf{G}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathsf{G}_{\text{SPEC}}}(1^\lambda) = 1]\right|$*,*

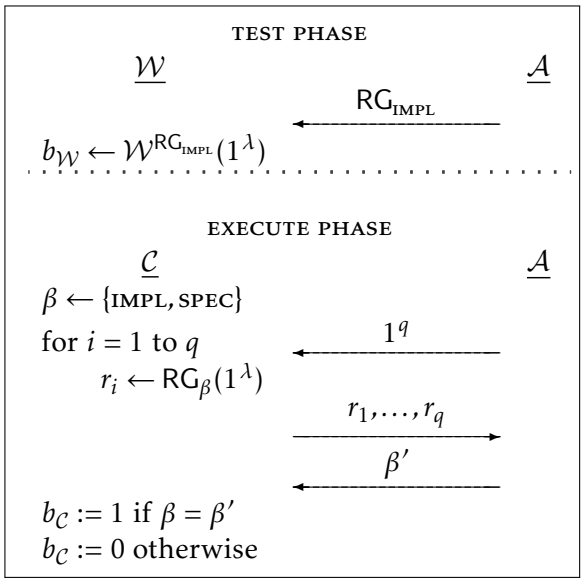


Figure 12: Game for stego-free randomness generation with trusted amalgamation.

A.3 Subversion resistant symmetric-key encryption with long messages

Definition A.3. *For any randomized symmetric-key encryption scheme, consider a specification* $\mathbf{E}_{\text{SPEC}} := (\mathsf{KG}_{\text{SPEC}}, \mathsf{Enc}_{\text{SPEC}}, \mathsf{Dec}_{\text{SPEC}})$. *We say specification* $\mathbf{E}_{\text{SPEC}}$ *is* subversion resistant with a trusted addition and amalgamation in the offline watchdog model, *if there exists an offline* PPT *watchdog* $\mathcal{W}$, *for any* PPT *adversary* $\mathcal{A}$, *playing the game defined in Figure 14, either,*

$$\mathbf{Adv}_{\mathcal{A}} \text{ is negligible, or, } \mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible.}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr[b_{\mathcal{C}} = 1] - \frac{1}{2}|$, *and* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = |\Pr[\mathcal{W}^{\mathbf{E}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathbf{E}_{\text{SPEC}}}(1^\lambda) = 1]|$,

A.4 Subversion resistant public key encryption with long messages

Definition A.4. *For any public key encryption scheme, consider a specification* $\mathbf{E}_{\text{SPEC}} := (\mathsf{KG}_{\text{SPEC}}, \mathsf{Enc}_{\text{SPEC}}, \mathsf{Dec}_{\text{SPEC}})$. *We say specification* $\mathbf{E}_{\text{SPEC}}$ *is* subversion resistant with a trusted addition and amalgamation in the offline watchdog model, *if there exists an offline* PPT *watchdog* $\mathcal{W}$, *for any* PPT *adversary* $\mathcal{A}$, *such that, at least one of the following conditions hold in the game defined in Figure 15:*

$$\mathbf{Det}_{\mathcal{W},\mathcal{A}} \text{ is non-negligible, or, } \mathbf{Adv}_{\mathcal{A}} \text{ is negligible,}$$

where $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr[b_{\mathcal{C}} = 1] - \frac{1}{2}|$, *and* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda) = |\Pr[\mathcal{W}^{\mathbf{E}_{\text{IMPL}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathbf{E}_{\text{SPEC}}}(1^\lambda) = 1]|$.

B Purifying randomness in the standard model

The split-and-amalgamate paradigm proposed in Section 3.2 assumes the specification of the immunization function Φ_{SPEC} to be modeled as a random oracle. Intuitively, it seems hard to remove this assumption when the randomness generation is split into only two segments, since

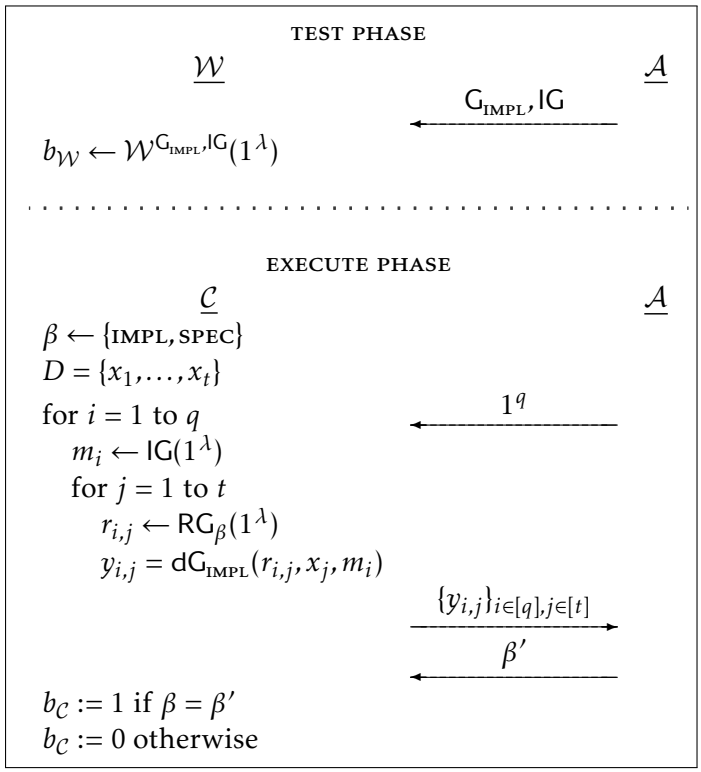


Figure 13: Game for stego-free randomized algorithm with trusted amalgamation. $\mathsf{IG}_{\text{IMPL}}$ is the instance generator, and D is a poly-size input domain.

a PPT offline watchdog can guarantee at most a $\omega(\log \lambda)$ entropy in the output of each $\mathsf{RG}^b_{\text{IMPL}}$ for $b = 0, 1$, (and the concrete lower bound of entropy is not even clear). Nevertheless, it is interesting to consider immunization in the standard model if we are willing to have more than constant number of components.

A simple approach via multi-splitting. Observe that suppose the randomness generation implemented by the adversary only outputs one bit, the watchdog (who checks whether there is a bit appears significantly more frequently than the other after drawing certain amount of samples) can guarantee that the bit only has a negligible bias. If not, suppose the bias for outputting bit value 1 is $1/2 + \delta$ for a noticeable δ, then according to the Chernoff bound, after the watchdog draws enough samples, he will notice that there are significantly more 1s appear which is abnormal if the specification is used which outputs a uniform bit.

With the above observation, we can simply extract one bit from a $\mathsf{RG}^1_{\text{IMPL}}$ from each draw. If we now split the randomness generation into n copies $\mathsf{RG}^1_{\text{IMPL}}, \ldots, \mathsf{RG}^n_{\text{IMPL}}$, we can collect n-bits if we independently draw one bit from each of them.

Security analysis. We now analyze how these n-bits are distributed. Suppose the output distribution of $\mathsf{RG}^i_{\text{IMPL}}$ is denoted by D_i, the user first samples $d_i \leftarrow D_i$, and outputs only the first bit of d_i; each such bit has bias at most $1/2 + \epsilon$, for a negligible function ϵ. We denote each bit using the random

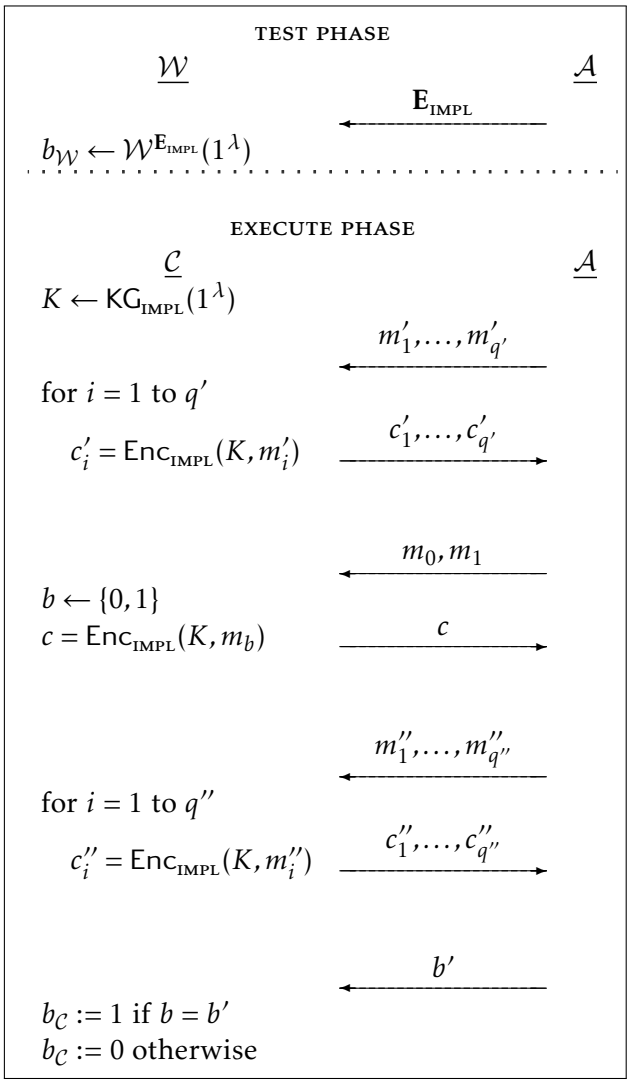


Figure 14: Subversion-resistant symmetric-key encryption with trusted addition.

variable $X_1, \ldots, X_n$. It follows that for any particular value $b_1, \ldots, b_n \in \{0,1\}^n$, as $1 + \frac{x}{n})^n \leq e^x$ we have

$$\Pr[(X_1, \ldots, X_n) = (b_1, \ldots, b_n)] \leq (1/2 + \epsilon)^n \leq \frac{1}{2^n} e^{2n\epsilon},$$

Let U_n be the uniform distribution over $\{0,1\}^n$, and Y_n be the distribution of $X_1 \ldots X_n$. Recall that if $Y_n(\bar{x}) \leq U_n(\bar{x})(1+t)$ for all $\bar{x} \in \{0,1\}^n$, the statistical distance satisfies:

$$\|Y_n - U_n\| = \frac{1}{2} \sum_{\bar{x}} |Y_n(\bar{x}) - U_n(\bar{x})| \leq \frac{1}{2} \sum_{\bar{x}} U_n(\bar{x}) \cdot t = \frac{t}{2}.$$

Putting back the value $t = e^{2n\epsilon} - 1$, we conclude that the statistical distance between the distribution of $X_1 \ldots X_n$ and the uniform distribution over n-bit strings, is no more than $2n\epsilon \leq \mathsf{negl}(\lambda)$, the inequality holds because $e^x - 1 \leq 2x$ for $x < 1/2$. Summarizing above, we have:

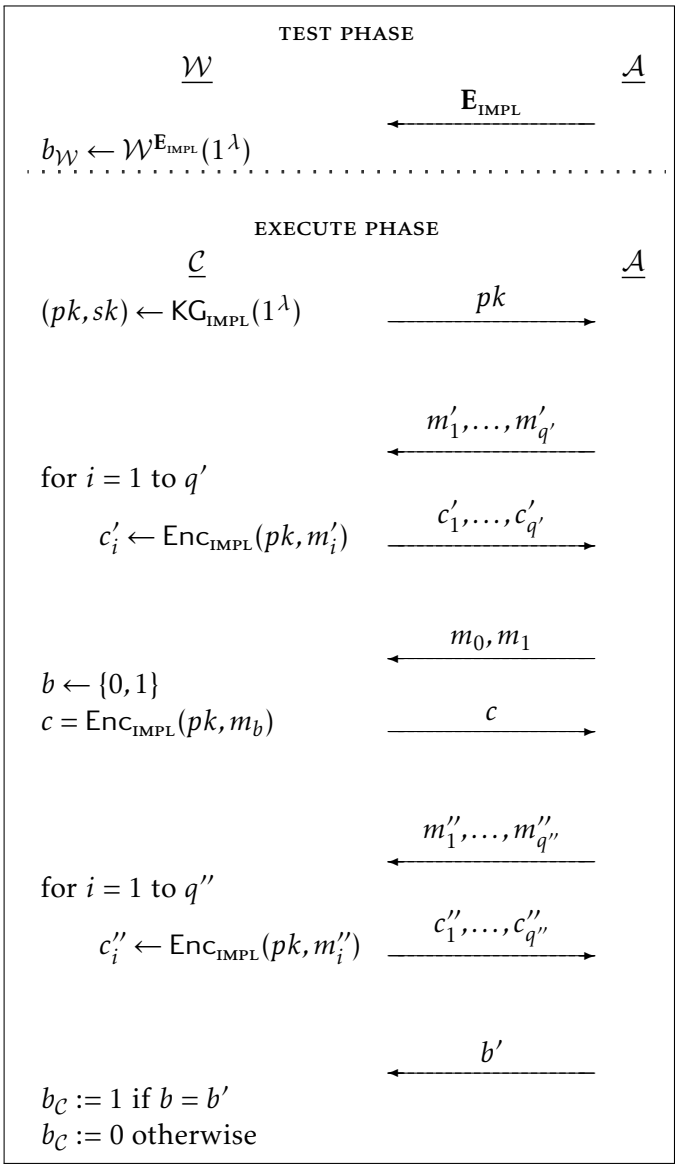


Figure 15: Subversion-resistant public key encryption supporting large messages.

Proposition B.1. *Suppose the randomness generation specification* $\mathsf{RG}_{\text{SPEC}}$ *is defined as* $(\mathsf{RG}^1_{\text{SPEC}}, \ldots, \mathsf{RG}^n_{\text{SPEC}})$*, where each* $\mathsf{RG}^i_{\text{SPEC}}$ *is supposed to output uniform n bits, and* $\Phi_{\text{SPEC}} : \{0,1\}^{n^2} \to \{0,1\}^n$ *simply outputs the first bit of each n-bit block. Such* $\mathsf{RG}_{\text{SPEC}}$ *is stego-free with a trusted amalgamation.*

Remark B.1. *We keep the immunization function for consistency with the general model. In this simple multi-splitting, we may not even need* Φ_{SPEC}*; we can define each* $\mathsf{RG}^i_{\text{SPEC}}$ *to output only 1 random bit.*

A more efficient construction. The above construction is extremely simple, but with a price that the randomness generation has to be split into n pieces if the output is n-bit long. Note that there are at least $c \log n$ bits entropy (for any constant $c \geq 2$) in the output of each $\mathsf{RG}^i_{\text{IMPL}}$. This comes from the guarantee of a watchdog that makes $O(n^{2c})$ queries during the interrogation. Moreover,

the bitstring can be stretched (if it is close to uniform) to a polynomially longer output using a PRG. [8]

We may use more powerful machineries of a strong randomness extractor together with a pseudorandom generator. The intuition is to first get a short seed and then use the power of the extractor to bootstrap the randomness generation by a factor of $\log n$ since we can get at least this many uniform bits from the output of each $\mathsf{RG}^i_{\mathrm{IMPL}}$. After collecting $O(n^\epsilon)$ bits (for some small ϵ), we can apply the PRG to stretch it to $O(n)$ bits. Suppose $\mathsf{RG}_{\mathrm{SPEC}}$ now is defined as $(\mathsf{RG}^1_{\mathrm{SPEC}}, \ldots, \mathsf{RG}^\ell_{\mathrm{SPEC}}, \Phi_{\mathrm{SPEC}})$, where $\ell = n^\epsilon/\log n + \log n$ for some small ϵ. We describe the immunization function $\Phi_{\mathrm{SPEC}} : \{0,1\}^{n\cdot\ell} \to \{0,1\}^n$ below, when inputing $r_1, \ldots, r_\ell$ (generated by $\mathsf{RG}^i_{\mathrm{IMPL}}$ respectively).

- First, Φ_{SPEC} takes $r_1, \ldots, r_{\log n}$ and returns the first bit of each r_i and obtains a $\log n$-bit uniform string denoted by s_0.
- Then, Φ_{SPEC} applies a seeded strong extractor Ext to $r_{\log n+1}, \ldots, r_\ell$ respectively using s_0 as a seed. It obtains $\ell \log n = n^\epsilon$ uniform bits, ($\log n$ bits from each of them), which are denoted by s_1.
- Last, we apply a pseudorandom generator PRG on s_1 to stretch it to n bits, and output those as the final random coin.

Security analysis. Let us first assume Φ_{SPEC} is not subverted. In the first step, s_0 would be close to uniform due to Proposition B.1. In the second step, since Ext is a strong extractor, the output of $\mathsf{Ext}(s_0, r_i)$ is close to uniform and independent of s_0, via the simple union bound, s_1 is also negligibly close to uniform. Last step follows easily because of the PRG property.

Next, observe that although Φ_{SPEC} here is a complex function, it is still a deterministic algorithm, and it is with a public input distribution. Following Lemma 2.2, the watchdog can guarantee that the subverted implementation Φ_{IMPL} will be consistent with the specification with an overwhelming probability when the input is sampled from $\mathsf{RG}^1_{\mathrm{IMPL}} \times \ldots \times \mathsf{RG}^\ell_{\mathrm{IMPL}}$.

Proposition B.2. *There exists a specification for the randomness generation that outputs n bits is stego-free with the trusted amalgamation and $O(n^\epsilon/\log n)$ segments for any constant ϵ.*

C Further Applications

We show above how we can salvage randomized algorithms, in particular randomized encryption schemes, when they are under subversion. While the technique we develop is used to generically eliminate subliminal channels which has its own interest. In this section, we give two more examples to illustrate how the techniques can be applied to broader settings. We first show how to circumvent the impossibility result of [DGG+15] for publicly immunizing the *output* of a potentially subverted PRG (called backdoored PRG). Then we show how our decomposition-then-amalgamate paradigm can be used to simulate an intuitively more powerful semi-private model proposed in [DGG+15]. In such a model, there is a uniformly random string r generated by a trusted party and r is made public only after the adversary provides the implementations. This will motivate follow-up works to consider the power of the semi-private model, then realize it using our paradigm.

[8] We remark that here the PRG is just one deterministic algorithm, and it is associated with a public input distribution. Such PRG can be based on e.g., some *fixed* hard function. This is in contrast with the (backdoored) PRG that requires some randomly generated public parameters PP, and has the danger of containing backdoors in PP [DGG+15, RTYZ15].

Public immunization of output of a backdoored PRG. In a backdoored PRG, the stretch algorithm takes a random seed and also a public key (generated by the adversary) as inputs. The goal of the adversary is to ensure that the output of the PRG looks uniform to everyone except herself by providing a malicious public key. In particular, it captured the notorious subversion attack on the Dual_EC PRG. Unfortunately, as shown in [DGG+15], publicly immunizing the outputs of the backdoored PRG using a random oracle is impossible, i.e., the adversary can still distinguish the immunized string from a random string. The reason is similar to the attack shown in Section 3.1 since the subverted stretch algorithm can make random oracle queries during the execution. Russell et al. proposed a way to sidestep this impossibility by randomizing the public key [RTYZ15] with the caveat that the specification of the key generation should output uniform group elements. As pointed out by [DGG+15], immunizing the output of the PRG is preferable as one does not have to pay particular attention to the key/parameter generation.

It is not hard to see our split-and-amalgamate paradigm can be applied here to immunize the *output* of a backdoored PRG. We present a simplified version that the PRG is stateless, it is not hard generalize it to allow iterations to capture the Dual_EC PRG, and we defer that to the full version. Moreover, if we allow to split into more pieces, we can immunize the output in the standard model. Even in the semi-private model with a trusted randomness, the known result uses either random oracle or UCE, neither of which can be explicitly instantiated so far.

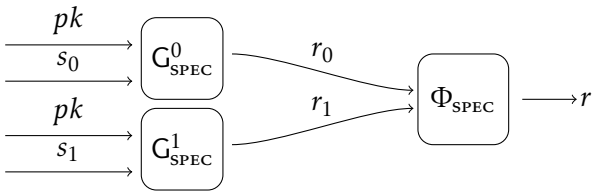


Figure 16: Immunizing backdoored PRG.

Security analysis. It is almost the same as the analysis of Theorem 3.2, if we can show (i) For each s_b, it is unpredictable to $\mathsf{G}^{1-b}_{\text{IMPL}}$; (ii) $\mathsf{G}_{\text{IMPL}}(s_0)\|\mathsf{G}_{\text{IMPL}}(s_1)$ is unpredictable to the adversary who makes the implementations. This is easy to see since the watchdog can check the entropy contained in the output r_0, r_1 respectively by feeding PRG with random seed and pk. We defer the proof to the full version.

Simulating the semi-private model. Semi-private model was introduced in [DGG+15], where a uniformly random string r is generated by a trusted party and r is made public only after the big brother provides the implementations. It was shown in [DGG+15] that in such a model, one can bypass the impossibility of public immunization of backdoored PRG output. We may expect more interesting results in this model due to the power of this extra trusted randomness.

Using our technique, we could ask the adversary to provide a stego-free specification of randomness generation, for example, we may ask the adversary to implement $\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}, \Phi_{\text{IMPL}}$, and the user can use them to generate bits that look uniform even to the adversary. In this way, we can simulate the semi-private model.

D Omitted Proofs

Proposition 3.1. *For any CPA-secure (public-key) encryption scheme, for any public function Φ, the adversary $\mathcal{A}$ shown in Figure 4 can learn the plaintext with probability 1 given the ciphertext generated by $\mathsf{Enc}^z_{\mathrm{IMPL}}$ even if the randomness generation is separated and immunized by a random oracle. Furthermore, suppose $\mathsf{RG}_{\mathrm{IMPL}}$ outputs ℓ bits of randomness; then the detection advantage is $q^2/2^{\ell-4} + \mathsf{negl}(\lambda)$ for all PPT watchdogs making q queries during the interrogation, assuming PRF is a secure pseudorandom function.*

Proof. The effectiveness of $\mathcal{A}$ is straightforward, since the randomness generated by $\mathsf{RG}_{\mathrm{IMPL}}$ makes the ciphertext to be distinguishable using the PRF, thus the adversary $\mathcal{A}$ who knows the backdoor recovers the message bit perfectly.

Next, we will argue that no offline watchdog can notice the subversion, particularly $\mathsf{RG}_{\mathrm{IMPL}}$. (All other components are honestly implemented). We define the game sequence as follows:

Game-0 is the game that $\mathcal{W}$ is interacting with $\mathsf{RG}_{\mathrm{IMPL}}$.

Game-1 is the same as Game-0 except that the PRF used in $\mathsf{RG}_{\mathrm{IMPL}}$ is replaced with a random function R.

Game-2 is the same as Game-1 except that $\mathsf{RG}_{\mathrm{IMPL}}$ resamples, if $\mathcal{W}$ notices a collision in the responses to the q queries.

Game-3 is the same as Game-2 except that $\mathsf{RG}_{\mathrm{IMPL}}$ is replaced with $\mathsf{RG}_{\mathrm{SPEC}}$.

Game-4 is the same as Game-3 except removing the resampling condition.

Analyzing the gaps, we see: Game-0 and Game-1 are indistinguishable because of the PRF security; Game-1 and Game-2 are identical, if there is no collision among the q queries in Game-2. In Game 2, for any r_0, the probability that $R(c_0) = 0 \wedge R(c_1) = 1$ is 1/4, where c_0, c_1 are ciphertexts encrypting $0, 1$ respectively using $\Phi(r_0)$ as the coin. Suppose S is the set that contains all the values that satisfies $R(c_0) = 0 \wedge R(c_1) = 1$, then the expected size of S, $E[|S|] = 2^\ell/4 = 2^{\ell-2}$. It follows that with negligible probability, $|S| \leq \frac{E[|S|]}{2} = 2^{\ell-3}$. Then the probability that there exists a collision among q samples from S would be bounded by $q^2/|S| \leq q^2/2^{\ell-3}$. Game-2 and Game-3 is identical, since the responses can either appear in a random subset or the whole space. Game-3 and Game-4 are identical except there is collision when sampling q uniform points. The probability of such collision exist is $q^2/2^\ell$. Combining them above, we have the proposition. □

Theorem 3.4. *For any randomized algorithm G, consider the specification $\mathsf{G}_{\mathrm{SPEC}} := (\mathsf{RG}_{\mathrm{SPEC}}, \mathsf{dG}_{\mathrm{SPEC}})$, where $\mathsf{RG}_{\mathrm{SPEC}}$ generates λ bits of uniform randomness and $\mathsf{dG}_{\mathrm{SPEC}}$ is deterministic. Let $(\mathsf{RG}^0_{\mathrm{SPEC}}, \mathsf{RG}^1_{\mathrm{SPEC}}, \Phi_{\mathrm{SPEC}})$ be the double-split specification of $\mathsf{RG}_{\mathrm{SPEC}}$ as above. If (1) $\mathsf{RG}^0_{\mathrm{SPEC}}$ and $\mathsf{RG}^1_{\mathrm{SPEC}}$ output λ uniform bits; (2) Φ_{SPEC} takes $r_0 \circ r_1$ as input, and outputs r; (3) $\mathsf{dG}_{\mathrm{SPEC}}$ is deterministic and it takes r and x as input, where x is generated by a sampler $\mathsf{IG}_{\mathrm{IMPL}}$ (see Fig. 6), then $\mathsf{G}_{\mathrm{SPEC}}$ is stego-free with a trusted amalgamation (according to Def. A.2 in appendix. A.2). Here Φ_{SPEC} is modeled as a random oracle, and $\Phi^0_{\mathrm{IMPL}}, \Phi^1_{\mathrm{IMPL}}$ are executed independently.*

Proof. The watchdogs will be a combination of the ones in Theorem 3.2 and Lemma 2.2 to guarantee unpredictability of $\mathsf{RG}^b_{\mathrm{IMPL}}$ and the overwhelming consistency for deterministic algorithms with a public input distribution. Here $\mathsf{dG}_{\mathrm{SPEC}}$ is a deterministic algorithm and the output of $\mathsf{RG}_{\mathrm{SPEC}} \times \mathsf{IG}_{\mathrm{IMPL}} \times D$ would be the input distribution to $\mathsf{dG}_{\mathrm{SPEC}}$.

We here only describe briefly about the game changes.

Game-0 is the same as Figure 13, the adversary $\mathcal{A}$ prepares every piece of the implementation, and the challenger simply calls them and passes the inputs to the next implementation as defined (doing the basic amalgamation).

Game-1 is the same as Game-0 except that the randomness r is uniformly sampled by $\mathcal{C}$.

Game-2 is the same as Game-1 except that $\mathsf{dG}_{\text{IMPL}}$ is replaced with $\mathsf{dG}_{\text{SPEC}}$.

Note that in Game-0, it corresponds to $b = 0$, while in Game-2, every implementation of the algorithm (except input generation) is now the specification, it corresponds to $b = 1$.

From Theorem 3.2, with a trusted amalgamation, the output from the implementation $\mathsf{RG}_{\text{IMPL}} := (\mathsf{RG}^0_{\text{IMPL}}, \mathsf{RG}^1_{\text{IMPL}}, \Phi_{\text{IMPL}})$ is pseudorandom to the adversary $\mathcal{A}$ who made $\mathsf{RG}_{\text{IMPL}}$. Thus in the security game defined in Figure 13, we can let the challenger simply generate r uniformly to reach Game-1.

From Lemma 2.2, $\mathsf{dG}_{\text{SPEC}}$ will be a deterministic algorithm with a public input distribution, thus $\mathsf{dG}_{\text{IMPL}}$ would be consistent with $\mathsf{dG}_{\text{SPEC}}$ with an overwhelming probability when inputs are sampled according to $\mathsf{RG}_{\text{SPEC}} \times \mathsf{IG}_{\text{IMPL}} \times D$, thus Game-2 can be reached with only a negligible gap.

Once in Game-2, all components are specification. □