Let a Non-Barking Watchdog Bite: Cliptographic Signatures with an Offline Watchdog

Sherman S. M. Chow[1⋆], Alexander Russell[2⋆⋆], Qiang Tang[3⋆⋆⋆], Moti Yung[4], Yongjun Zhao[5], and Hong-Sheng Zhou[6†]

[1] Chinese University of Hong Kong, sherman@ie.cuhk.edu.hk
[2] University of Connecticut, acr@cse.uconn.edu
[3] New Jersey Institute of Technology, qiang@njit.edu
[4] Google Inc. and Columbia University, moti@cs.columbia.edu
[5] Chinese University of Hong Kong, yjzhao@ie.cuhk.edu.hk
[6] Virginia Commonwealth University, hszhou@vcu.edu

Abstract. We study how to construct secure digital signature schemes in the presence of kleptographic attacks. Our work utilizes an offline watchdog to clip the power of subversions via only one-time black-box testing of the implementation. Previous results essentially rely on an online watchdog which requires the collection of all communicating transcripts (or active re-randomization of messages).

We first give a simple but generic construction, without random oracles, in the partial-subversion model in which key generation and signing algorithms can be subverted. Then, we give the first digital signature scheme in the complete-subversion model in which all cryptographic algorithms can be subverted. This construction is based on the full-domain hash. Along the way, we enhance the recent result of Russell *et al.* (CRYPTO 2018) about correcting a subverted random oracle.

Keywords: Signatures, Subversion Resilience, Offline Watchdog

1 Introduction

Modern cryptography has been spectacularly successful. We have already seen a flurry of cryptographic tools with versatile functionalities and rigorous security analyses. Yet, the formal security guarantees come with an implicit caveat – they only hold if the implementations faithfully realize the specifications the formal security proof is analyzing. Our experiences tell us that implementation can be tricky. Programming bugs may go undetected and subtle errors can make the implementation faulty. Apart from unintended blunders which may spoil the security guarantee, implementations of cryptographic algorithms can be subverted

⋆ Supported by GRF (CUHK 14210217) of the Research Grants Council, Hong Kong
⋆⋆ Supported in part by NSF award 1801487
⋆⋆⋆ Supported in part by NSF award 1801492
† Supported in part by NSF award 1801470

with *fully adversarial implementations* which look correct even under fairly intensive (black-box) testing. Such kind of subversion, or in general, *kleptographic* attack [27,28], is not just a pathological concern, but has been understood as a real threat since the Snowden revelations [22]. Concretely speaking, whenever a "third-party" software library or hardware device is relied upon by a bigger cryptographic system, it is hard to assert its security even if the said cryptographic system is "provably secure" in the traditional sense.

At a high level, kleptography considers a "proud-but-curious" adversary whose goal is to break the security of a certain cryptographic primitive by supplying a malformed implementation of it without being detected. Under such a setting, the adversary has many viable attack strategies. For example, the malicious implementation of a signature verification algorithm may always return "1" when seeing a certain hard-coded string[7]. For another example, the subverted randomized (*e.g.*, encryption) algorithm may leak secret information via a steganographic channel [4,23]. These general and powerful attack strategies are undetectable under offline black-box testing. Given these attacks, it is not surprising that all existing defense mechanisms rely on extra trust assumptions, such as trusted online reverse firewall [21,16,10], trusted key generation algorithm [5,2], trusted initialization [17], *etc.*

Recently, Russell *et al.* [23] proposed a framework (called cliptography) for systematically studying how to secure cryptographic primitives in the presence of kleptographic attacks, *i.e.*, how to clip the power of kleptographic attacks. The framework is characterized by three parties: an adversary, who may provide potentially subverted implementations of cryptographic algorithms; a "watchdog", who either certifies or rejects the implementations by subjecting them to (black-box) interrogation according to the genuine specification of the algorithms; and a challenger, who plays with the adversary in a conventional security game, but now using the potentially subverted implementations. This framework is capable of capturing a wide range of subversion capabilities and defense mechanisms.

- **Online watchdog vs. offline watchdog.** We can define two flavors of watchdogs, depending on the information given to it. The strong (and less attractive) model of *online watchdog* [23] is provided with access to the full transcript of the challenger-adversary security game. It could be valuable for establishing feasibility results, but in practice, it is not easy to instantiate such a watchdog, as it has to piggyback on the implementations, collecting all communication transcripts to detect abnormal inputs, and "barking" all the time. The weaker (and perhaps more attractive) model is the *offline watchdog* model [23,25]. The watchdog simply interrogates the supplied implementations, comparing them with the specification of the primitives, and declares them to be "fit" or "unfit." In other words, the watchdog only needs to "bark" once, and then it can go offline afterward.
- **Partial subversion vs. complete subversion.** The adversary may be more interested in subverting certain cryptographic algorithms than the others. For instance, if the attack goal is to learn the secret signing key, the

[7] This can be viewed as applying the input-triggered attack [13] to signature schemes.

attacker will be less interested in subverting the verification algorithm than the signing algorithm, since the verification result can only carry 1-bit of information and it is likely to be kept locally. It is thus still worthy to consider the *partial-subversion* model, in which some algorithms can be explicitly excluded from subversion in the security game. Some subversion defense methods are established in this model, *e.g.*, honest key generation algorithm and honest verification algorithm of a digital signature scheme [2]. The cliptography framework by Russell *et al.* [23] can easily capture partial subversion by letting the challenger run the genuine algorithm in the security game. Of course, it is of great importance to consider a more powerful adversary who can launch a *complete subversion* which subverts *all* the relevant cryptographic elements of a scheme [23] (excluding the computing base).

– **Trusted computing base.** Note that the complete-subversion model above only refers to the *functional components*, *i.e.*, the cryptographic algorithms, which should be distinguished from the user computing base for basic operations such as $\oplus$, $=$, "reassembly", *etc.* The trusted computing base for (some of) these operations is provided by the architecture, which is normally not under the control of the cryptography implementation/library provider.

  Russell *et al.* [23,25] recently proposed the split-program strategy for immunizing kleptographic attacks on randomized algorithms. The idea of this non-black-box technique is to decompose the algorithm into a *constant* number of smaller components. The adversary can still provide subverted implementations of *all* these components but the challenger will faithfully *amalgamate* these components into a fully functional implementation, which will be used in the security game. Note that all components are still subject to black-box interrogation by the (online/offline) watchdog. Such non-black-box testing and trusted amalgamation can be captured by simply providing specifications of all small components of the algorithm to the watchdog.

Current status of subversion-resistant signatures. To the best of our knowledge, only three previous works considered subversion-resistant signature schemes. The work by Ateniese *et al.* [2] not only relies on a priori "verifiability" condition which essentially requires an online watchdog to instantiate, but also assumes trusted key generation algorithm (or requires a trusted online "reverse firewall"). The result of Russell *et al.* [23], despite in the complete-subversion model, (explicitly) requires an online watchdog too. Fischlin and Mazaheri [17] recently proposed a new defense mechanism called "self-guarding" which requires users to have a trusted initialization phase to generate genuine message signature pairs for randomly chosen messages. We continue the pursuit of reducing the trust assumption needed for subversion-resistant signature schemes.

1.1 Our Contributions

We investigate subversion-resistant EUF-CMA-secure (simply put, cliptographic) digital signature schemes in the above framework with only an offline watchdog.

A simple generic construction in the partial-subversion model. We start with a simple construction which works for any existing signature schemes. So, one can just apply a simple "patch", or install a "small" (due to its simplicity) add-on without changing the underlying system. Note that for this generic construction, the verification algorithm is still trusted.

How difficult is our problem? First, note that the *key generation* can be handled by the recent double-splitting technique [25]. The main difficulty appears in the Sign algorithm. Recall that one potential catastrophe of a subverted signing algorithm is the revelation of the secret key. It is relatively easier to discover such a subversion if the secret key is blatantly output as the "signature". A more sophisticated kleptographic attacker will hide this secret. When the signing algorithm is randomized, it provides a convenient subliminal channel. A natural preventative measure is to use clean randomness to re-randomize the signatures (if they are publicly re-randomizable), with the existence of a cryptographic reverse firewall [21]. Alternatively, a *unique* signature scheme, in which there is only one valid signature for each message, simply does not feature any subliminal channel. These explain in a high-level way the feasibility results of Ateniese *et al.* [2]. Nevertheless, many signature schemes, especially those efficient ones with security proven in the standard model, are randomized (*e.g.* [8]). So our first question is: can we upgrade the signing algorithm of a probabilistic signature scheme?

Our generic construction. A general defense against input-triggered attacks [13] is to mandate that the subverted implementation only takes a random message. Russell *et al.* [25] construct subversion-resistant encryption schemes in the offline-watchdog model with this idea. The encryption algorithm invokes two instances of encryption, one encrypting $u \oplus m$ and the other encrypting u. Adopting this strategy naïvely in the context of signature signing does not work. The scheme $\mathsf{Sign}_{\text{SPEC}}(\mathsf{sk}, m) = (\mathsf{Sign}'_{\text{SPEC}}(\mathsf{sk}, u), \mathsf{Sign}'_{\text{SPEC}}(\mathsf{sk}, u \oplus m), u)$ is trivially forgeable.

We fixed this forgeable scheme by two techniques: i) Domain-separation: We append different special symbols to the inputs in the two invocations, so that the output of the first invocation cannot be interpreted as the second one (and vice versa); ii) One-time random tag: We also need to make sure that no one can mix-and-match (the components of) signatures for different messages to create new forgeries. To do so, we further include a random tag r that binds the two signature components together, also making sure that they are one-use only. We note that the domain-separation technique has been used in other contexts, such as random oracle instantiation. Similar one-time random tag structure has also appeared in the context of structure-preserving signature [1], but their work does not randomize the message to be signed.

Moreover, to handle the subliminal channel attack due to biased randomness, we decouple the randomness generation from the randomized algorithm [25]. The randomness generation can be further handled via the double-splitting technique, while the deterministic counterpart for signing can be safeguarded by only an offline watchdog as we feed only uniform messages as input.

FDH-based construction in the complete-subversion model. Our main contribution is a secure signature scheme in the complete-subversion model which further handles the subverted verification algorithm.[8] This is the first signature scheme that achieves such security goals. The simple generic construction above cannot handle subversion of verification algorithm. Indeed, it is not clear how to generically apply the randomization strategy to the potentially adversarial inputs to be fed to the verification (*i.e.*, the message m and the signature σ), such that the signature verification algorithm still works on these randomized inputs, without jeopardizing the unforgeability of the signature scheme.

Our second construction hence does not take the generic randomization approach, but instead handles the classical full-domain hash (FDH) [7,11] paradigm. In this paradigm, the signing algorithm first hashes the message and then inverts the hashed value via a trapdoor one-way permutation. The adversary is supposed to provide the implementation of each algorithm: $\mathsf{KG}_{\text{IMPL}}$, $\mathsf{Sign}_{\text{IMPL}}$, $\mathsf{Verify}_{\text{IMPL}}$ and also the implementation of the hash h_{IMPL}.

First, we note that the *key generation* can be handled the same way as our generic transformation above applying the recent double-splitting technique [25].

Regarding the *hash function*, we utilize the recent work of Russell *et al.* [26], which provides a simple construction that can correct a subverted random oracle, such that the resulting function will be as good as an ideal random function. The construction requires some public randomness that is generated after the implementation of the hash is supplied. To apply their theorems [26], and ensure that the "corrected" hash can be considered to be a random oracle, we need to ensure i) the subversion disagrees on its specification only at a negligible fraction; ii) there is randomness that can be generated and published after the malicious implementations are supplied; iii) "interpret" their "replacement lemma" [26] such that it is suitable for our application. Point iii) is more complex than it looks, especially when all the other algorithms are subverted. See below.

It is challenging to deal with the *signing algorithm*. To avoid the signing implementation to leak the secret triggered by some hidden input, we will apply the "corrected" random oracle [26] to the message before passing it into the evaluation function of the underlying one-way permutation. The adversary is required to provide the implementation of the inversion function, and the implementation of the hash, separately, to enforce the actual inputs to the implementation of inversion function are $\mathsf{sk}, \tilde{h}_R(m)$, which are generated by a known distribution.

However, we remark that simply viewing $\tilde{h}_R$ as a good random oracle $g(\cdot)$ (trivially applying the replacement lemma [26]) is still problematic. As the subverted $\mathsf{Inv}_{\text{IMPL}}$ could simply use $g(z)$ as the backdoor and output the secret key sk directly when z appears in a signing query (*i.e.*, $\mathsf{Sign}_{\text{IMPL}}(\mathsf{sk}, z) = \mathsf{Inv}_{\text{IMPL}}(\mathsf{sk}, g(z)) = \mathsf{sk}$).

[8] As elaborated above, the trusted computing base including operations like "$\oplus$" and "$=$" are still in place. They are actually necessary due to the known (simple) trigger attacks [13] assuming only an offline watchdog. Our goal is to reduce the number of trusted functional components, and keep the remaining as simple as possible, *e.g.*, without any trusted large group operations.

The problem here is that the adversary can query random oracle when generating the implementations and plant the trigger accordingly. To defend against such attack, we have to disable the adversary from making useful random oracle queries during the implementation-generation phase. Observe that if we have some randomness R generated after $\mathsf{Sign}_{\text{IMPL}}$ is provided, and R is involved in the "encoding" of the message before sending to $\mathsf{Inv}_{\text{IMPL}}$, then the above problem could be mitigated. Luckily, the correction function from [26] already involves randomness generated after the time that implementations are provided. What we need to adapt here is to derive a "stronger replacement theorem" that the correction function of [26] is actually "as good as" (in the sense of indifferentiability) a *keyed* hash (where, the key could be public, but sampled after the implementation is provided). See Section 4.3 for details.

Finally, it is also tricky to deal with the *verification algorithm*. Suppose the implementation of the verification takes input public key pk and a message-signature pair (m, σ), and outputs 0 or 1 to decide whether the signature is valid. The input-triggered attack again can be applied here in a way that, for some randomly chosen message m^*, $\mathsf{Verify}_{\text{IMPL}}(\cdot, m^*, \cdot)$ always outputs 1. Opening up the verification functionality of the full-domain hash signature, it is actually to check whether evaluating the signature equals to the ("corrected") hash of the message. We propose to do such canonical verification explicitly, that the equality operation (and the "corrected" hash) will be done by the user. The adversary will provide the implementation of the evaluation function. This simple decomposition of the verification functionality changes the task of the adversarial implementation from targeting one bit to predicting a random value, which is the output of the "corrected" hash. We remark here that, as above, the use of the public randomness is also critical to prevent the adversary from making useful random oracle queries during the manufacturing phase of $\mathsf{Verify}_{\text{IMPL}}$.

There still exists a subtler attack, that the attacker might use the trigger signature material σ^* to directly carry the information of $h_R(m^*)$. This has to be resolved by strictly restricting the length of σ^* and doing a length check. As σ^* first needs to carry certain trigger information which is independent of the output of $h_R(m^*)$, this thus burns the information needed for a precise prediction of the value of $h_R(m^*)$.

1.2 Related Works

Kleptography introduced by Young and Yung [27,28] primarily highlighted the possibility of subverting key generation and left open the problem of defending against such subversion. A recent line of work of Russell *et al.* [23,25,26] has initiated a systematic study of cliptography about defending against kleptographic attacks by redesigning the specification and leveraging architectural tools. In particular, they provided a subversion-resistant digital signature, assuming an online watchdog [23].

Also recently, new attacks and defense mechanisms in the kleptographic setting keep appearing. In particular, Bellare *et al.* [5] studied subverted randomized encryption algorithms, building a steganographic channel that leaks secrets bit

by bit. Indeed, subliminal channel attacks turn out to be the major obstacle in this area, and have been further explored by Ateniese *et al.* [2], Bellare *et al.* [3,4], Degabriele *et al.* [13], Dodis *et al.* [15], and Liu *et al.* [19]. A common feature of these works [5,3,4,13] is to adopt *deterministic* algorithms and to assume honest key generation to defend against subliminal channel attacks.

Furthermore, these works do not rely merely on testing. In fact, most 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 the work of Degabriele *et al.* [13]. However, it relies on an online watchdog that possesses access to the actual challenger-adversary communication transcript (including the internal state of the challenger).

Another research line [21,16,10] considered defense mechanisms with a "reverse firewall" that faithfully "re-randomizes" incoming and outgoing communication. On one hand, this model is attractive as it may permit quite general feasibility results. On the other hand, it relies on an independent component which is a source of trusted randomness (which generalized the "trusted warden" [14] used to eliminate subliminal channels in authentication protocols) and "re-randomizable" structure of the underlying algorithms.

Recently, Fischlin and Mazaheri [17] proposed a new defense mechanism called "self-guarding", which assumes that a genuine version of the cryptographic implementations is available before they get substituted. The self-guarding primitive then leverages information gathered using that genuine implementation at the initial phase to re-randomize potentially malicious inputs like the reverse firewall approach (assuming trusted basic operations like exclusive-or or group operation). They constructed several self-guarding primitives including digital signature schemes. Besides the trusted "setup", their signature construction comes at a price that verification/signing key size and signature size all inflate by a factor of $O(\lambda)$ where λ is the security parameter.

Finally, also motivated by the doubt on the implementation, cryptographers (*e.g.*, [29,18]) studied combiners of cryptographic primitives such that as long there exists one component primitive is secure, even if it is not known which one is that, the combined primitive remains secure.

Organization. In Section 2, we define the security for subversion-resistant digital signature. In Section 3, we give our first construction – a simple and generic scheme in the partial-subversion model; in Section 4, we give our second construction – an FDH-based signature scheme in the complete-subversion model. Both constructions use only an offline watchdog. Finally, the crooked indifferentiability model can be found at Appendix A.

2 Definition of Subversion-Resistant Signatures

First, we recall the definition of subversion-resistant signatures [23]. Its goal is fairly simple: the security of the digital signature scheme – unforgeability – should be preserved even one uses the malicious implementations supplied by

the adversary, as long as the adversarial implementation is not detected. The detection is done by a trusted entity called watchdog who has the specification of the algorithms and it will interrogate (via oracle accesses of) the implementation to see whether it is consistent with the specification. The subversion-resistant signature game is defined as the classical unforgeability game, except that the challenger will use the *implementations supplied by the adversary* instead of the specification of the algorithms. In particular, the challenger runs the key generation algorithm $\mathsf{KG}_{\text{IMPL}}$ to generate the challenged signing key and verification key, uses the signing functionality $\mathsf{Sign}_{\text{IMPL}}$ to answer signing queries and use the implementation of verification functionality $\mathsf{Verify}_{\text{IMPL}}$ to verify the final forgery that the adversary made. Definition 1 formalizes the high-level description above. It can be viewed as a special case of the cliptographic game [23, Definition 2] under the context of digital signature schemes.

Definition 1. *A specification* $\Pi_{\text{SPEC}} = (\mathsf{KG}_{\text{SPEC}}, \mathsf{Sign}_{\text{SPEC}}, \mathsf{Verify}_{\text{SPEC}})$ *for a digital signature scheme* Π *is* **subversion-resistant** *in the offline-watchdog model, if there exists a probabilistic polynomial-time (*PPT*) watchdog* $\mathcal{W}$*, s.t., for any* PPT *adversary* $\mathcal{A}$ *playing the security game (Figure 1) with the challenger* $\mathcal{C}$*, either the* advantage *of the adversary* $\mathcal{A}$ *in the security game* $\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \Pr[b_{\mathcal{C}} = 1]$ *is negligible, or the* detection probability $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda)$ *of the watchdog* $\mathcal{W}$ *with respect to* $\mathcal{A}$ *is non-negligible. Here,* $\mathbf{Det}_{\mathcal{W},\mathcal{A}}(1^\lambda)$ *is defined by*

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

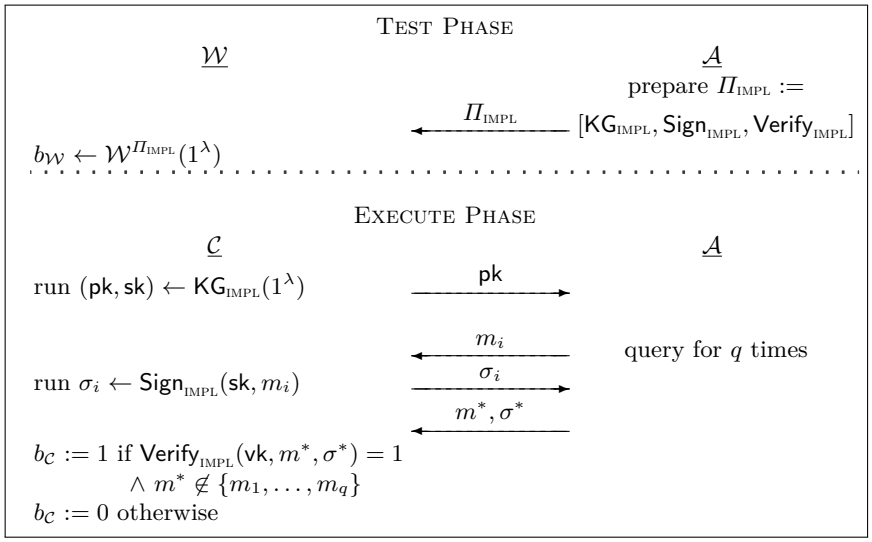

Fig. 1. Subversion-Resistant Signature Game in the Offline-Watchdog Model

As discussed earlier in Section 1, depending on the watchdog power, there could be different variants of the above model. The most realistic watchdog only

performs one-time testing, which is called an offline watchdog. In practice, an offline watchdog can be some industrial labs or security experts. We can also consider a more stringent *online* watchdog that additionally checks all communication transcripts between the challenger and the adversary. The online-watchdog model has been explicitly considered under the context of digital signatures [23]. Clearly, an online watchdog is much powerful and makes the design of subversion-resistant scheme easier, but it is also more costly to realize an online watchdog. An online watchdog has to piggyback on the implementation and actively monitor all communications of an implementation.

Unfortunately, with only an offline watchdog, it is impossible to achieve unforgeability in the kleptographic setting [2], even if only the Sign algorithm is subverted. To see, recall the input-triggered attack mentioned above: the subverted signing algorithm $\mathsf{Sign}_{\text{IMPL}}$ simply outputs the secret key when signing on a hard-coded trigger message m which is selected uniformly by the adversary. It is obvious that the adversary can make one single signing query to totally break the unforgeability. Previous work [23] got around this by introducing an online watchdog. Another work [2] introduced a "verifiability" assumption – any signature generated by the malicious signing implementation should be verified by the genuine Verify algorithm. This verifiability assumption can only be ensured with an online watchdog. This impossibility holds when the implementation is used as a black box, without doing any post-processing. We will show below that if the user can do some basic operation, *e.g.*, equality check and "$\oplus$", then it is possible to construct a digital signature scheme secure against the powerful kleptographic attack, with only an offline watchdog that performs non-black-box testing (*i.e.*, trusted amalgamation).

3 A Simple Generic Construction in the Standard Model

We propose a generic transformation on the signing algorithm which leads us to a new randomized subversion-resistant signature scheme in the offline-watchdog model from any deterministic signature scheme that is existentially unforgeable against adaptive chosen-message attack ($\mathsf{EUF\text{-}CMA}$, *cf.*, Definition 8) (assuming trusted verification and "$\oplus$"). Our transformation (modulo the underlying algorithms) holds in the standard model[9], and can be easily generalized to handle randomized signatures as well. As discussed in Section 2, previous subversion-resistant signature schemes either rely on an online watchdog [23], or an online reverse firewall [10], or a strong "verifiability" assumption [2].

Figure 2 below formally describes our construction. For the sake of simplicity, we describe the transformation for deterministic signature schemes first, and then show how to generalize the result to handle randomized schemes.

[9] In the full version ([24]) of [25], the authors discussed how to achieve subversion-resistant randomness generation in the standard model, at the cost of efficiency. See Appendix B and [24] for details.

Key generation. We handle the key generation by adopting the recently proposed double-splitting technique [25, Theorem 3.5], which we recall in Appendix B. This guarantees that the implementation of a carefully designed specification of key generation can be used as good as the specification, as long as the randomness generation algorithm is executed independently. We refer to [25] for details. Our result can be lifted to allow malicious key generation by directly applying the existing technique [25].

Sign. We augment the specification of the signing algorithm $\mathsf{dSign}_{\text{SPEC}}$ with a random tag generator $\mathsf{RG}_{\text{SPEC}}$ and a random message generator $\mathsf{MG}_{\text{SPEC}}$, *i.e.*, $\mathsf{Sign}_{\text{SPEC}} = (\mathsf{RG}_{\text{SPEC}}, \mathsf{MG}_{\text{SPEC}}, \mathsf{dSign}_{\text{SPEC}})$. $\mathsf{RG}_{\text{SPEC}}$ and $\mathsf{MG}_{\text{SPEC}}$ are merely for generating uniformly random tags and messages of a certain length. Therefore, they can also be handled[10] by the double-splitting technique [25, Theorem 3.4], similar to the key generation algorithm. To sign a message m, the user first runs $\mathsf{MG}_{\text{IMPL}}$ (the implementation) to sample a random message u, and compute a message $m' = u \oplus m$. The user also runs $\mathsf{RG}_{\text{IMPL}}$ to generate a random tag r from some super-polynomial-size domain. The user will call $\mathsf{dSign}_{\text{IMPL}}$ twice to sign two distinct messages $m_1 = (r||u||\text{“1”})$ and $m_2 = (r||m'||\text{“2”}) = (r||u \oplus m||\text{“2”})$, where “1” and “2” are two special symbols. The ultimate output of the signing algorithm is $\sigma = (r, u, \sigma_1, \sigma_2)$ where σ_1, σ_2 are the corresponding output of the two invocations of $\mathsf{dSign}_{\text{IMPL}}$.

Verify. Verification works straightforwardly: parse σ as $(r, u, \sigma_1, \sigma_2)$, compute $m' = u \oplus m$, compose m_1 and m_2 (using trusted “$\oplus$”), and verify σ_1 and σ_2.

Given an EUF-CMA-secure deterministic signature $\mathcal{SS}'_{\text{SPEC}} := (\mathsf{KGen}'_{\text{SPEC}}, \mathsf{dSign}'_{\text{SPEC}}, \mathsf{Verify}'_{\text{SPEC}})$, and assuming trusted “$\oplus$”, our subversion-resistant signature scheme $\mathcal{SS}_{\text{SPEC}} := (\mathsf{KGen}_{\text{SPEC}}, \mathsf{Sign}_{\text{SPEC}}, \mathsf{Verify}_{\text{SPEC}})$ is defined below:

- Key generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\text{SPEC}}(\lambda)$, where $\mathsf{KGen}_{\text{SPEC}}(\lambda)$ is the stego-free version of $\mathsf{KGen}'_{\text{SPEC}}$ in the trusted-amalgamation model (see Theorem 5 [25, Theorem 3.5] in Appendix B).
- Sign: $\sigma \leftarrow \mathsf{Sign}_{\text{IMPL}}(\mathsf{pk}, \mathsf{sk}, m)$, where $\mathsf{Sign}_{\text{SPEC}}(\mathsf{pk}, \mathsf{sk}, m)$ is given by:
  sample uniformly random string and message $r \leftarrow \mathsf{RG}_{\text{SPEC}}(1^\lambda)$, $u \leftarrow \mathsf{MG}_{\text{SPEC}}(1^\lambda)$, where $\mathsf{RG}_{\text{SPEC}}$ and $\mathsf{MG}_{\text{SPEC}}$ are stego-free randomness generation algorithms (see Theorem 4 [25, Theorem 3.4] in Appendix B);
  compute $\sigma_1 \leftarrow \mathsf{dSign}'_{\text{SPEC}}(\mathsf{sk}, (r||u||\text{“1”}))$ and $\sigma_2 \leftarrow \mathsf{dSign}'_{\text{SPEC}}(\mathsf{sk}, (r||u \oplus m||\text{“2”}))$;
  output $\sigma = (r, u, \sigma_1, \sigma_2)$.
- Verification: $b \leftarrow \mathsf{Verify}_{\text{SPEC}}(\mathsf{pk}, m, \sigma)$, where $\mathsf{Verify}_{\text{SPEC}}$ is given by:
  parse the input σ as $(r, u, \sigma_1, \sigma_2)$;
  run $\mathsf{Verify}'_{\text{SPEC}}(\mathsf{pk}, (r||u||\text{“1”}), \sigma_1)$ and $\mathsf{Verify}'_{\text{SPEC}}(\mathsf{pk}, (r||u \oplus m||\text{“2”}), \sigma_2)$;
  return 1 if and only if both verifications succeed.

Fig. 2. Subversion-Resistant Signature Scheme $\mathcal{SS}_{\text{SPEC}}$ in the Offline-Watchdog Model

[10] $\mathsf{RG}_{\text{SPEC}}$ and $\mathsf{MG}_{\text{SPEC}}$ will be split into three pieces exactly in Figure 14.

Before detailing the security analysis, we briefly explain how our design ensures security. First, $\mathsf{KG}_{\text{SPEC}}$ is subversion resistant because we are directly applying the result in [25] (also see Theorem 5). Second, the $\mathsf{Sign}_{\text{SPEC}}$ algorithm is subversion resistant because by design the input to $\mathsf{dSign}'_{\text{SPEC}}$ comes from a public (uniform) distribution. A simple watchdog can further guarantee that $\mathsf{dSign}_{\text{IMPL}}$ is consistent with the specification when the output is sampled from $\mathcal{SK} \times \mathcal{R} \times \mathcal{M} \times \{\text{“1”}, \text{“2”}\}$, where $\mathcal{SK}$ denotes the space of signing keys, $\mathcal{R}$ denotes the super-polynomial-size tag space, and $\mathcal{M}$ denotes the message space. Third, the special symbols (“1” and “2”) and the random tag r ensure EUF-CMA-security as follows: (1) the two special symbols separate the input domain so that the output of the first invocation of $\mathsf{dSign}'_{\text{SPEC}}$ (with “1” appended) cannot be the output of the second invocation (with “2” appended) for a forgery of $\mathsf{Sign}_{\text{SPEC}}$, and vice versa; (2) the random tag r drawn from a super-polynomial-size domain makes sure that the signature $\sigma = (\sigma_1, \sigma_2)$ for some message m is one-use only: the adversary cannot mix-and-match different signatures to create new forgeries.

Theorem 1. *For any* EUF-CMA*-secure deterministic digital signature scheme* $\mathcal{SS}'_{\text{SPEC}}$*, the specification* $\mathcal{SS}_{\text{SPEC}}$ *described in Figure 2 is subversion resistant in the trusted-amalgamation model, assuming a trusted “*$\oplus$*” operation and trusted the verification algorithm, and* $\mathsf{RG}_{\text{SPEC}}$ *outputs uniformly random tag from some super-polynomial-size domain.*

Proof. The watchdog for $\mathcal{SS}_{\text{SPEC}}$ is a combination of the watchdogs of the underlying components, including watchdogs for key generation ($\mathsf{KG}_{\text{SPEC}}$), random tag generation ($\mathsf{RG}_{\text{SPEC}}$), random message generation ($\mathsf{MG}_{\text{SPEC}}$). There is also a watchdog that makes sure $\mathsf{dSign}_{\text{IMPL}}$ is consistent with the specification on inputs sampled from $\mathcal{SK} \times \mathcal{R} \times \mathcal{M} \times \{\text{“1”}\}$ and $\mathcal{SK} \times \mathcal{R} \times \mathcal{M} \times \{\text{“2”}\}$ (*cf.* Theorem 7) because these two distributions are both public.

To see, the inputs to $\mathsf{dSign}'_{\text{SPEC}}$ consist of the signing key sk from $\mathsf{KGen}_{\text{IMPL}}$ and the concatenation of the following: a uniformly random tag r from $\mathsf{RG}_{\text{IMPL}}$, a uniform message u or $u \oplus m$ (where u comes from $\mathsf{MG}_{\text{IMPL}}$), and a special symbol (“1” or “2”). With the trusted “$\oplus$” operation, $u \oplus m$ will look uniform to $\mathsf{dSign}'_{\text{IMPL}}$, hence its distribution is also $\mathsf{MG}_{\text{IMPL}}$.

The rest of the proof consists of two parts. The first part is a series of game transitions showing that from the adversary's point of view, the subversion game (as defined in Figure 1) is indistinguishable from a standard EUF-CMA game (namely $\mathcal{SS}_{\text{IMPL}}$ is replaced by $\mathcal{SS}_{\text{SPEC}}$), conditioned on the event that the watchdogs above do not detect any abnormal behavior of $\mathcal{SS}_{\text{IMPL}}$. The second part shows that $\mathcal{SS}_{\text{SPEC}}$ is indeed EUF-CMA-secure, so that the adversary's advantage is indeed negligible.

Now we sketch the game changes and explain the negligible differences arise during a series of game transitions from $\mathcal{SS}_{\text{IMPL}}$ to $\mathcal{SS}_{\text{SPEC}}$, conditioned on the watchdog's result ($\mathsf{Verify}_{\text{IMPL}}$ is assumed trusted). Let the advantage of the adversary $\mathcal{A}$ in game G_i be $\mathbf{Adv}^{G_i}_{\mathcal{A}}$.

Game-0. G_0 is the original game as described in Figure 1 with a trusted amalgamation.

Game-1. G_1 is identical to G_0 except that the key generation implementation $\mathsf{KG}_{\text{IMPL}}$ is replaced by its specification $\mathsf{KG}_{\text{SPEC}}$.

Lemma 1. $|\mathbf{Adv}^{G_0}_{\mathcal{A}} - \mathbf{Adv}^{G_1}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. This lemma follows straightforwardly from Theorem 5 [25, Theorem 3.5]. Namely, if KGen is split into $\mathsf{RG}_0, \mathsf{RG}_1, \Phi, \mathsf{dKG}$, and $\mathsf{RG}_0, \mathsf{RG}_1$ are executed independently, the resulting implementation would be stego-free, *i.e.*, indistinguishable from the specifications even to the adversary. (The formal definition of stego-freeness is recalled in Appendix B. Readers are referred to [25] for a more detailed discussion.) □

Game-2. G_2 is the same as G_1 except $\mathsf{MG}_{\text{IMPL}}$ is replaced by $\mathsf{MG}_{\text{SPEC}}$.

Game-3. G_3 is the same as G_2 except $\mathsf{RG}_{\text{IMPL}}$ is replaced by $\mathsf{RG}_{\text{SPEC}}$.

Lemma 2. $|\mathbf{Adv}^{G_2}_{\mathcal{A}} - \mathbf{Adv}^{G_1}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$ *and* $|\mathbf{Adv}^{G_3}_{\mathcal{A}} - \mathbf{Adv}^{G_2}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. These two inequalities follow directly from Theorem 4 [25, Theorem 3.4]. Taking the first inequality as an example, if there exists an adversary $\mathcal{A}$ such that $|\mathbf{Adv}^{G_2}_{\mathcal{A}} - \mathbf{Adv}^{G_1}_{\mathcal{A}}|$ is non-negligible, we can build another adversary $\mathcal{B}$ breaking stego-freeness game for $\mathsf{MG}_{\text{IMPL}}$. The reduction is straightforward: $\mathcal{B}$ simulates the rest of the game for $\mathcal{A}$, receives all malicious implementations ($\mathsf{MG}_{\text{IMPL}}$, $\mathsf{dSign}_{\text{IMPL}}$, *etc.*), but forwards only $\mathsf{MG}_{\text{IMPL}}$ to its own watchdog and challenger. Whenever $\mathcal{A}$ queries the signing oracle for some m_i, $\mathcal{B}$ asks its own challenger to obtain u_i generated either by $\mathsf{MG}_{\text{IMPL}}$ or by $\mathsf{MG}_{\text{SPEC}}$, and uses u_i to compute appropriate responses for $\mathcal{A}$ using implementations provided by $\mathcal{A}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. It is easy to see that the simulation is perfect, and the advantage of $\mathcal{B}$ is the same as the advantage of $\mathcal{A}$. The second inequality follows the same argument. □

Game-4. G_4 is the same as G_3 except $\mathsf{dSign}'_{\text{IMPL}}$ is replaced by $\mathsf{dSign}'_{\text{SPEC}}$. Note that in G_4, all the implementations have been replaced by their genuine specifications.

Lemma 3. $|\mathbf{Adv}^{G_4}_{\mathcal{A}} - \mathbf{Adv}^{G_3}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. This again follows from Theorem 5 using a similar argument as in Lemma 1. Note that the inputs to dSign' are drawn from public distributions (either $\mathcal{SK} \times \mathcal{R} \times \mathcal{M} \times \{\text{“1”}\}$ or $\mathcal{SK} \times \mathcal{R} \times \mathcal{M} \times \{\text{“2”}\}$). □

Finally, we need to show that $\mathbf{Adv}^{G_4}_{\mathcal{A}}$ is indeed negligible, which is equivalent to showing that $\mathcal{SS}_{\text{SPEC}}$ is indeed an EUF-CMA-secure signature scheme. To this end, we design a simple reduction algorithm reducing the EUF-CMA-security of $\mathsf{Sign}_{\text{SPEC}}$ to that of $\mathsf{dSign}'_{\text{SPEC}}$. Suppose there is an adversary $\mathcal{A}$ that breaks EUF-CMA-security of $\mathsf{Sign}_{\text{SPEC}}$, we design an adversary $\mathcal{B}$ that breaks $\mathsf{dSign}'_{\text{SPEC}}$. For any signing query m, $\mathcal{B}$ randomly chooses (r, u), and submits signing queries $(r||u||\text{“1”})$ and $(r||u \oplus m||\text{“2”})$ to the oracle $\mathcal{O}^{\mathsf{dSign}'_{\text{SPEC}}}(\cdot)$. $\mathcal{B}$ locally maintains a

list of records in the form $(r, u, m, \sigma_1, \sigma_2)$ where σ_1, σ_2 are the responses from $\mathcal{O}^{\mathsf{dSign}'_{\text{SPEC}}}(\cdot)$, and forwards $\sigma = (r, u, \sigma_1, \sigma_2)$ to $\mathcal{A}$. Eventually $\mathcal{A}$ outputs a forgery (σ^*, m^*), where $\sigma^* = (r^*, u^*, \sigma_1^*, \sigma_2^*)$, with non-negligible probability.

To see how $\mathcal{B}$ can extract a valid forgery for $\mathsf{dSign}'_{\text{SPEC}}$ from $\mathcal{A}$'s forgery (σ^*, m^*), notice that by (σ^*, m^*) being a valid forgery for $\mathsf{Sign}_{\text{SPEC}}$, it means that both $(\sigma_1^*, (r^*||u^*||\text{“1”}))$ and $(\sigma_2^*, (r^*||u^* \oplus m^*||\text{“2”}))$ are valid message-signature pairs for $\mathsf{dSign}'_{\text{SPEC}}$, and that $\mathcal{A}$ has never queried the signing oracle for m^*. The latter indicates that $\mathcal{B}$'s local list does not contain any entry in the form $(\cdot, \cdot, m^*, \cdot, \cdot)$. We discuss several cases:

1. Tag r^* does not appear in any entries of $\mathcal{B}$'s record: Both $(\sigma_1^*, (r^*||u^*||\text{“1”}))$ and $(\sigma_2^*, (r^*||u^* \oplus m^*||\text{“2”}))$ are valid forgeries for $\mathsf{dSign}'_{\text{SPEC}}$;
2. Tag r^* exists in some records in the form $(r^*, u, \cdot, \cdot, \cdot)$: By our assumption that the tags are supposed to be drawn from a super-polynomial-size space, with overwhelming probability all the tags in $\mathcal{B}$'s record are unique. Without loss of generality, assume that this unique record is $(r^*, u, m, \sigma_1, \sigma_2)$. That means $(r^*||u||\text{“1”})$ and $(r^*||u \oplus m||\text{“2”})$ are the only queries sent to $\mathcal{O}^{\mathsf{dSign}'_{\text{SPEC}}}(\cdot)$ which begin with tag r^*. If $u^* \neq u$, then $(\sigma_1^*, (r^*||u^*||\text{“1”}))$ must be a valid forgery for $\mathsf{dSign}'_{\text{SPEC}}$;
3. r^* and u^* appear in a unique entry of the form $(r^*, u^*, m, \sigma_1, \sigma_2)$: Using the same argument above, $(r^*||u^*||\text{“1”})$ and $(r^*||u^* \oplus m||\text{“2”})$ are the only queries sent to $\mathcal{O}^{\mathsf{dSign}'_{\text{SPEC}}}(\cdot)$ which begin with tag r^*. It must hold that $m^* \neq m$. Otherwise, $\mathcal{A}$ must have asked for a signature for m^* from $\mathcal{B}$. Given that $m^* \neq m$, $(\sigma_2^*, (r^*||u^* \oplus m^*||\text{“2”}))$ must be a valid forgery for $\mathsf{dSign}'_{\text{SPEC}}$.

By our assumption that $\mathsf{dSign}'_{\text{SPEC}}$ is EUF-CMA-secure, $\mathbf{Adv}_{\mathcal{A}}^{G_4}$ is negligible. Putting all the lemmas above together, we complete our proof. □

It is straightforward to generalize Theorem 1 to handle randomized signatures. Basically, the randomized signing algorithm $\mathsf{rSign}_{\text{SPEC}}(\mathsf{sk}, m)$ needs to be split into two components $\mathsf{RG}'_{\text{SPEC}}(1^\lambda)$ and $\mathsf{dSign}_{\text{SPEC}}(r; (\mathsf{sk}, m))$. $\mathsf{RG}'_{\text{SPEC}}$ generates uniform randomness needed by $\mathsf{rSign}_{\text{SPEC}}$, and $\mathsf{dSign}_{\text{SPEC}}$ is a deterministic algorithm, so that for all sk, m, it holds that $\mathsf{rSign}_{\text{SPEC}}(\mathsf{sk}, m) = \mathsf{dSign}_{\text{SPEC}}(\mathsf{RG}'_{\text{SPEC}}(1^\lambda); (\mathsf{sk}, m))$. Both $\mathsf{RG}'_{\text{SPEC}}$ and $\mathsf{dSign}_{\text{SPEC}}$ can be made subversion-resistant easily. The security proof above only needs to be augmented with an additional hybrid game that replaces $\mathsf{RG}'_{\text{IMPL}}$ with $\mathsf{RG}'_{\text{SPEC}}$.

Corollary 1. *For any* EUF-CMA*-secure (randomized) digital signature scheme* $\mathcal{SS}'_{\text{SPEC}}$*, the specification* $\mathcal{SS}_{\text{SPEC}}$ *described above is subversion-resistant in the trusted-amalgamation model, assuming a trusted “*$\oplus$*” operation and trusted verification algorithm, and* $\mathsf{RG}_{\text{SPEC}}$ *outputs uniformly random tag from some super-polynomial-size domain.*

4 FDH-based Signatures under Complete Subversion

Now we describe our second construction of signature scheme which only requires an offline watchdog when all cryptographic algorithms $(\mathsf{KG}, \mathsf{Sign}, \mathsf{Verify})$ and the

hash functions are subjected to subversion. Our scheme follows the full-domain hash [7,11] paradigm, one of the most classical applications of random oracles.

4.1 High-Level Ideas

In an FDH-based signature scheme, the signing algorithm first hashes the message and then inverts the hashed value using a trapdoor one-way permutation. Suppose the adversary can subvert the implementation of each algorithm: $\mathsf{KG}_{\text{IMPL}}, \mathsf{Sign}_{\text{IMPL}}, \mathsf{Verify}_{\text{IMPL}}$ and also the implementation of the hash h_{IMPL}. Several natural questions arise. Let us examine the algorithms one by one.

As discussed in the introduction, we will handle those algorithms one by one. Here we elaborate a bit more. The intuition for defending against the trigger is that the Sign algorithm cannot be fed with a random message. Without the trusted re-randomization, our idea is to hash the message. While hashing alone does not resolve the problem as the trigger can be trivially propagated through the hash. One simple observation is to hash the message together with some random element that is not known to the attacker, *e.g.*, public-key material. Now, this naturally leads us to consider hash subversion.

Fortunately, Russell *et al.* [26] provided a simple construction that can correct a subverted random oracle, such that the resulting function will be as good as an ideal random function. To apply their theorems [26], we need to ensure i) the subversion disagrees with its specification only at a negligible fraction; ii) there is randomness that can be generated and published after the malicious implementations are supplied; iii) interpret the "replacement" lemma to be suitable for our application. Requirement (i) is easy and repeatedly used in the cliptography literature [23,25]. As the hash function is a deterministic function, the offline watchdog can simply evaluate the implementation and compare with the output of the specification. For (ii), observe that the implementation of key generation $\mathsf{KG}_{\text{IMPL}}$ will produce a public key, which should be unpredictable to the adversary (otherwise, the watchdog can keep sampling to find a collision to differentiate $\mathsf{KG}_{\text{IMPL}}$ from $\mathsf{KG}_{\text{SPEC}}$). It follows that if $\mathsf{KG}_{\text{IMPL}}$ can be treated as honestly generated (see above), we can extend the key generation to also output some randomness R which will be part of the public key. Requirement (iii) is a bit subtler. Simply replacing the corrected hash with a trusted random oracle is not enough. See the next point of subverting $\mathsf{Sign}_{\text{IMPL}}$ and Section 4.3 below.

The traditional implementation of the verification takes input public key pk and a message-signature pair (m, σ), and outputs 0 or 1 to decide whether the signature is valid. The input-triggered attack can be applied here easily. $\mathsf{Verify}_{\text{IMPL}}(\cdot, m^*, \cdot)$ can just always outputs 1 for some randomly chosen message m^* (or a special signature element σ^*). In the full-domain hash, opening up the verification functionality, it is actually to check whether evaluating the signature is equal to the ("corrected") hash of the message. We first propose to do such a canonical verification explicitly, that the equality operation will be done by the user. The adversary will provide the implementation of the evaluation function of the one-way permutation. This simple decomposition of the verification functionality changes the task of the adversarial implementation

from targeting one bit to predicting a random value, which is the output of the "corrected" hash. We remark here that, same as above, the use of the public randomness is also important for preventing the adversary from making useful random oracle queries during the manufacturing phase of $\mathsf{Verify}_{\text{IMPL}}$.

There still exists a subtler attack, that the attacker might use the trigger signature material σ^* to directly carry the information of $h_R(m^*)$. Such kind of attack looks like the "big brother" $\mathcal{A}$ is communicating directly to the "little brother" – the subverted implementation for action items. This has to be resolved by strictly restricting the length of σ^* and doing a length check. We note that in the setting of FDH, since we use trapdoor one-way *permutation*, thus the length is tight, and the simple length checking already works. See the proof of Lemma 7.

4.2 Our Subversion-Resistant FDH-based Signature Scheme in the Offline-Watchdog Model

Given a trapdoor one-way permutation, with specification denoted by $\mathcal{F}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Eval}^{\mathcal{F}}_{\text{SPEC}}, \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}})$, and a public hash function (or a family of hash functions for consistency) with specification $\left\{h_i : \{0,1\}^* \to \{0,1\}^n\right\}_{i=0,\ldots,\ell}$, where we assume the message space is $\mathcal{M} = \{0,1\}^n$; we construct a subversion-resistant signature scheme $\mathcal{SS}$ with specification $\mathcal{SS}_{\text{SPEC}} := (\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}, \mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}, \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}})$.

Note that the family of $\{h_i\}_{i=1}^{\ell}$ may be simply derived from one hash using different indices, *e.g.*, $\forall x, h_i(x) = h(i,x)$, where $i = 1, \ldots, \ell = 3n+1$.

- Key generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}(\lambda)$, where $\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}$ [11] is given by:
  The algorithm generates $(f, td_f) \leftarrow \mathsf{KG}^{\mathcal{F}}_{\text{SPEC}}(\lambda)$, and $R := r_1, \ldots, r_\ell \leftarrow \{0,1\}^{n\ell}$.
  The algorithm sets $\mathsf{pk} := (f, R)$ and $\mathsf{sk} := td_f$;
- Sign: $\sigma \leftarrow \mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}}(\mathsf{pk}, \mathsf{sk}, m)$, where $\mathsf{Sign}^{\mathcal{SS}}_{\text{SPEC}} := (\{h_i\}_{i=0}^{\ell}, \mathsf{Inv}_{\text{SPEC}})$ is given by:
  Upon receiving message m, the algorithm first computes $\tilde{m} = h_R(m) = h_0\left(\bigoplus_{i=1}^{\ell} h_i(m \oplus r_i)\right)$, and then generate the signature as $\sigma = \mathsf{Inv}^{\mathcal{F}}_{\text{SPEC}}(\mathsf{sk}, \tilde{m})$.
  $\mathsf{Sign}_{\text{SPEC}} := (\{h_i\}_{i=1}^{\ell}, \mathsf{Inv}_{\text{SPEC}})$ means, explicitly, the adversary should follow this decomposition, and provide implementations of $\{\tilde{h}_i\}_{i=1}^{\ell}$ and $\mathsf{Inv}_{\text{IMPL}}$ individually.
- Verification: $b \leftarrow \mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}}(\mathsf{pk}, m, \sigma)$, where $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}} := (\{h_i\}_{i=1}^{\ell}, \mathsf{Eval}_{\text{SPEC}})$ is given by:
  Upon receiving message-signature pair (m, σ) and a public key pk, the algorithm only proceeds if the length of σ^* is correct (equals to the hash output length n), it then computes $\tilde{m} = h_R(m) = h_0\left(\bigoplus_{i=1}^{\ell} h_i(m \oplus r_i)\right)$, if $\mathsf{Eval}^{\mathcal{SS}}_{\text{SPEC}}(\mathsf{pk}, \sigma) = \tilde{m}$, set $b := 1$; otherwise, set $b := 0$. Here, $\mathsf{pk} = (f, R)$.
  Likewise, $\mathsf{Verify}^{\mathcal{SS}}_{\text{SPEC}} := (\{h_i\}_{i=1}^{\ell}, \mathsf{Eval}_{\text{SPEC}})$ means that for Verify the adversary should supply the implementation of $\mathsf{Eval}_{\text{IMPL}}$ (while $\{h_i\}$ can be reused).

[11] We remark here that the $\mathsf{KG}^{\mathcal{SS}}_{\text{SPEC}}$ algorithm will be split into four pieces exactly as [25].

4.3 How to use the Replacement Theorem [26]

To prepare us for the security proof, we first strengthen the previous result about correcting random oracle. Let us recall the replacement theorem [26] for establishing that a corrected random oracle is as good as a truly random function when used in larger systems.

General replacement with crooked indifferentiability. Security-preserving replacement has been shown in the indifferentiability framework [20]: if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem in the $\mathcal{G}$ model is at least as secure as that in the $\mathcal{F}$ model. It has been shown [26] that the replacement property can also hold in the *crooked indifferentiability* framework (see Appendix A.2 and [26] for a detailed definition).

To model "as secure" (when correcting a subverted object) when used in larger systems (see illustration in Figure 3 excluding R), consider an ideal primitive $\mathcal{G}$, we can define the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ as follows: Initially, the crooked environment $\widehat{\mathcal{E}}$ manufactures and then publishes a subverted implementation of $\mathcal{G}$, denoted by $\tilde{\mathcal{G}}$. Then $\widehat{\mathcal{E}}$ runs the attacker $\mathcal{A}$, and the cryptosystem $\mathcal{P}$ is developed. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle accesses to C whereas attacker $\mathcal{A}$ has oracle accesses to $\mathcal{G}$; note that, C has oracle accesses to $\tilde{\mathcal{G}}$, not directly to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle accesses to $\mathcal{F}$. Finally, the crooked environment $\widehat{\mathcal{E}}$ returns a binary decision output. It was shown [26] that if a construction C is $\mathcal{G}$-crooked indifferentiable with another object $\mathcal{F}$, $C^{\mathcal{G}}$ would be as secure as $\mathcal{F}$ when used in any larger system $\mathcal{P}$.

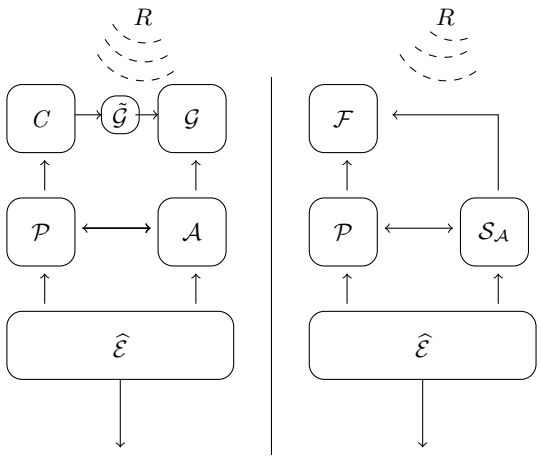

Fig. 3. Environment $\widehat{\mathcal{E}}$ interacts with Cryptosystem $\mathcal{P}$ and Attacker $\mathcal{A}$: In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle accesses to C whereas $\mathcal{A}$ has oracle accesses to $\mathcal{G}$; the algorithm C has oracle accesses to the subverted $\tilde{\mathcal{G}}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle accesses to $\mathcal{F}$.

*An **easier-to-use interpretation** for correcting subverted random oracles.* Using the definition and the theorem as is, however, will cause some trouble when applying the result of correcting a subverted random oracle, especially when plugging it to a larger system. We first reflect the public randomness generated after implementation is provided more explicitly in the model. Moreover, we also need to adjust the "ideal world" a little bit so that the targeted ideal object (in particular, a random oracle here) is also utilizing such public randomness, which yields a slightly stronger object of (ideal) *keyed* hash. These two adjustments will be critical for the application to our FDH construction.

For simplicity, we focus only on random oracles here. Consider a random oracle $\mathcal{G}$, we augment the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ as follows: Initially, the crooked environment $\widehat{\mathcal{E}}$ deploys the attacker $\mathcal{A}$ to query $\mathcal{G}$ for some preprocessing. It follows immediately $\widehat{\mathcal{E}}$ deploys the crooked implementation $\tilde{\mathcal{G}}$ and the cryptosystem $\mathcal{P}$ (which itself could be malicious or containing subverted components). Some randomness R is then drawn and published, which is utilized by construction C. On the other hand, in the world using random oracle $\mathcal{F}$, originally after R is generated, $\mathcal{F}(\cdot)$ becomes $\mathcal{F}(R,\cdot)$ (with the first half of inputs fixed by a randomly selected R). The interactions among $\mathcal{A}, \mathcal{P}, \mathcal{E}$ and the rest of the definition of "as secure" remain the same. See Figure 3.

Definition 2. *Consider random oracles $\mathcal{G}$ and $\mathcal{F}$ (both with variable input length). A cryptosystem $\mathcal{P}$ is said to be at least as secure in the augmented $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model, if for any augmented $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ and any attacker $\mathcal{A}$ in the augmented $\mathcal{G}$-crooked model, there exists an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\widehat{\mathcal{E}}(\mathcal{P}^{C^{\tilde{\mathcal{G}}}}, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\widehat{\mathcal{E}}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \leq \epsilon.$$

where ϵ is a negligible function of the security parameter λ.

We can prove a similar theorem as the replacement theorem [26] for the augmented definition (with essentially an identical proof technique, see the dashed frames in Figure 4 and we refer to [26] for details).

Corollary 2. *Let $\mathcal{P}$ be a cryptosystem with oracle accesses to a random oracle $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is $\mathcal{G}$-crooked-indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the augmented $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model.*

4.4 Security Analysis

Theorem 2. *If $\mathcal{F}_{\text{SPEC}}$ is a trapdoor permutation, the specification of $\{h_i\}_{i=0,\ldots,\ell}$ are random oracles, then the signature scheme $\mathcal{SS}$ with specification $\mathcal{SS}_{\text{SPEC}}$ constructed above is subversion resistant with an offline watchdog, assuming the "$\oplus$" and "$=$" operations are honestly carried out (and execute the pieces independently as [25]).*

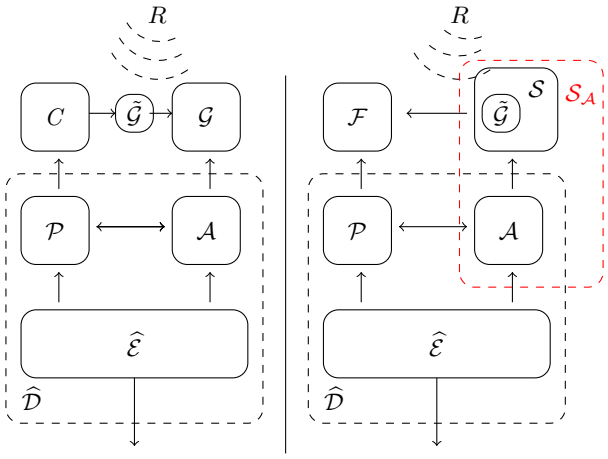

Fig. 4. Construction of Attacker $\mathcal{S}_{\mathcal{A}}$ from Attacker $\mathcal{A}$ and Simulator $\mathcal{S}$

Proof. First, to simplify the presentation of the analysis in the cliptographic setting, we ignore the checking phase of the offline watchdog in the game transitions, while taking the simple guarantees such as deterministic function will be correct on an overwhelming portion of inputs as the condition. The security can then be seen simply by walking through the sequence of game hopping over games G_i's (closer to the usual case). Let the advantage of adversary $\mathcal{A}$ in game G_i be $\mathbf{Adv}_{\mathcal{A}}^{G_i}$.

Game-0. G_0 is exactly the same security game as defined in Definition 1 (the execute phase with the challenger $\mathcal{C}$ using implementations provided by the adversary $\mathcal{A}$) instantiating with our construction described in Section 4.2. See Figure 5.

Game-1. G_1 is identical to G_0 except that the key generation implementation $\mathsf{KG}_{\text{IMPL}}$ is substituted with its specification $\mathsf{KG}_{\text{SPEC}}$. See Figure 6.

Lemma 4. $|\mathbf{Adv}_{\mathcal{A}}^{G_0} - \mathbf{Adv}_{\mathcal{A}}^{G_1}| \leq \mathsf{negl}(\lambda)$.

Proof. The proof is identical to the one for Lemma 1. □

Game-2. G_2 is identical to G_1 except that the message encoding function using corrected hash $\widetilde{h}_R(\cdot)$ is replaced with a truly random g parameterized by R, *i.e.*, $g(R, \cdot)$. See Figure 7.

Lemma 5. $|\mathbf{Adv}_{\mathcal{A}}^{G_1} - \mathbf{Adv}_{\mathcal{A}}^{G_2}| \leq \mathsf{negl}(\lambda)$.

Proof. This follows directly from Corollary 2 that the corrected function using subverted random oracle $\widetilde{h}_R(\cdot)$ can be replaced with a truly random function g indexed by the randomness R which is generated after.

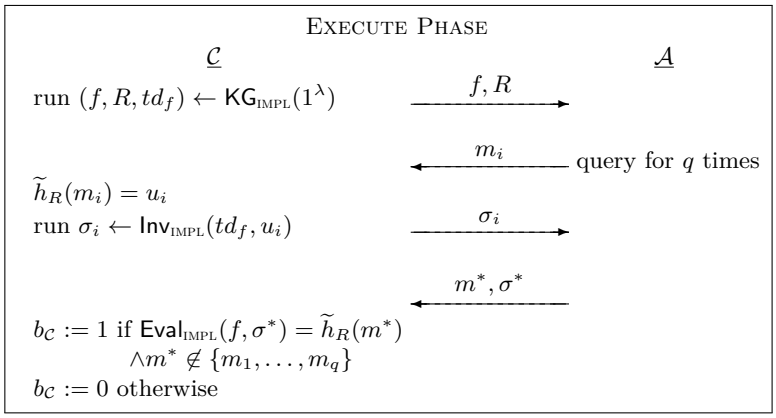

Fig. 5. Game-0: The original cliptographic signature game

We can simply view the augmented h-crooked environment in the corollary as the actual adversary here in the game, and the larger cryptosystem $\mathcal{P}$ is simply composed of the signature implementations. □

Game-3. G_3 is identical to G_2 except that the implementation of the actual signing function $\mathsf{Inv}_{\text{IMPL}}$ is substituted with its specification $\mathsf{Inv}_{\text{SPEC}}$. See Figure 8.

Lemma 6. $|\mathbf{Adv}^{G_2}_{\mathcal{A}} - \mathbf{Adv}^{G_3}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Now we need to demonstrate that when a keyed hash is used (the key is public but sampled after the implementation of the signing functionality is provided), $\mathsf{Inv}_{\text{SPEC}}$ is actually stego-free in the sense that the adversary cannot distinguish whether she is interacting with $\mathsf{Inv}_{\text{IMPL}}$ or $\mathsf{Inv}_{\text{SPEC}}$, even if she can freely choose potentially triggered inputs.

Let us first look at a simpler challenge game. Consider a random oracle $h : \{0,1\}^* \to \{0,1\}^n$. Suppose an attacker $\mathcal{A}$ makes some q_1 number of queries to h, define a target set $T \subset \{0,1\}^n$ with a polynomially large size q_2, generate uniform randomness R with length λ, and R public. The adversary will try to find an input x such that $h(R, x)$ falls into T. It is not hard to see that $\Pr[h(R, x) \in T] = \frac{q_1 q_2}{2^\lambda}$ which is negligible in λ if the adversary makes one attempt (and remains negligible if $\mathcal{A}$ makes polynomially many attempts).

Now instantiating such statement under our setting: simply using the points that $\mathsf{Inv}_{\text{IMPL}}$ differ from $\mathsf{Inv}_{\text{SPEC}}$ to define such T (the offline watchdog ensures that the "discrepancy set" T has to be exponentially small). Now when the adversary makes a signing query m, it is to find such an input that makes the output of $g(R, m)$ to fall into the target set T. This probability would be negligible. It follows that the output of $\mathsf{Inv}_{\text{IMPL}}$ and $\mathsf{Inv}_{\text{SPEC}}$ when evaluating on $g(R, x)$ will be the same for an overwhelming probability for every x. Thus $\mathsf{Inv}_{\text{IMPL}}$ satisfies the stego-free notion even with an adversarially chosen input x. □

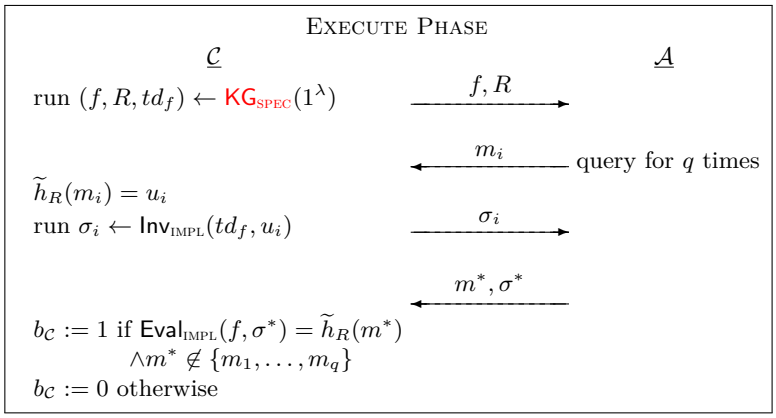

Fig. 6. Game-1: Honest key generation

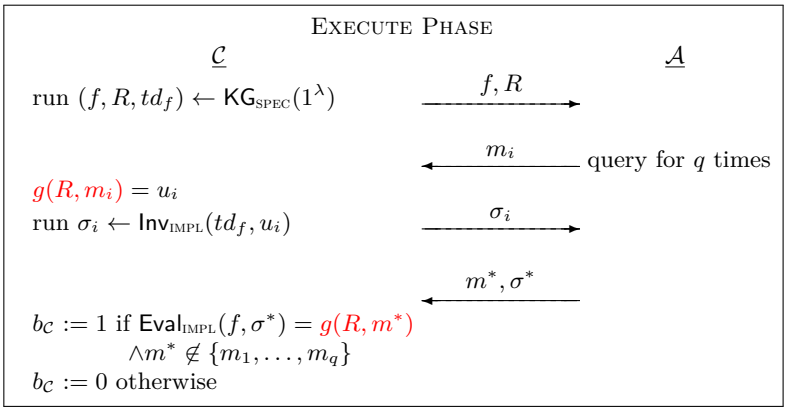

Fig. 7. Game-2: Corrected keyed hash

Game-4. G_4 is identical to G_3 except that the implementation of the actual verification function $\mathsf{Eval}_{\text{IMPL}}$ is substituted with its specification $\mathsf{Eval}_{\text{SPEC}}$. Now all the implementations are actually honestly generated, thus G_4 essentially falls back to the classical unforgeability game for FDH signatures. See Figure 9.

Lemma 7. $|\mathbf{Adv}^{G_3}_{\mathcal{A}} - \mathbf{Adv}^{G_4}_{\mathcal{A}}| \leq \mathsf{negl}(\lambda)$.

Proof. Now we need to demonstrate that when a keyed hash is used (the key is public but sampled after the implementation of the signing functionality is provided), and a trusted equality test is in place, $\mathsf{Eval}_{\text{IMPL}}$ performs essentially the same as $\mathsf{Eval}_{\text{SPEC}}$ when predicting an output of $g(R, m^*)$.

Suppose $\mathsf{Eval}_{\text{IMPL}}(f, \sigma^*) \neq \mathsf{Eval}_{\text{SPEC}}(f, \sigma^*)$, that means σ^* falls into the set of inputs that $\mathsf{Eval}_{\text{IMPL}}$ and $\mathsf{Eval}_{\text{SPEC}}$ differ. To escape from the watchdog's detection of this inconsistency, those inputs must contain at least $\omega(\lambda)$ bits of entropy

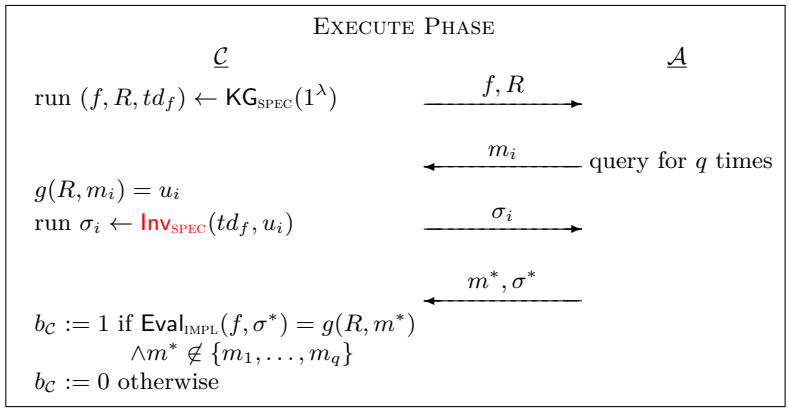

Fig. 8. Game-3: Honest Sign

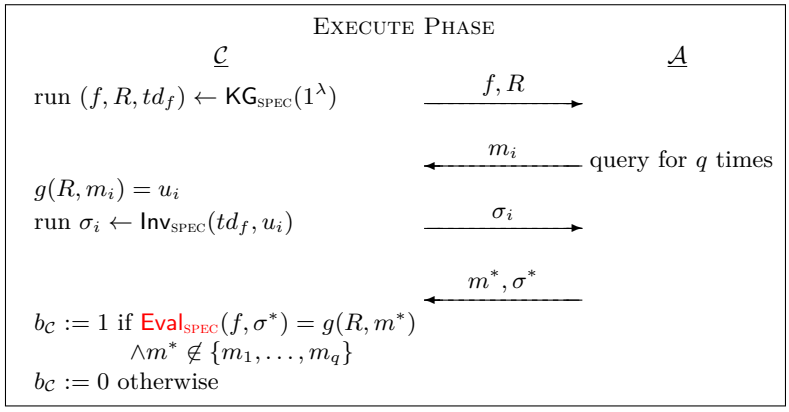

Fig. 9. Game-4: Honest Verify

about some trigger that $\mathsf{Eval}_{\text{IMPL}}$ can explore to recognize those inputs to deviate from the specification. Otherwise, the watchdog would be able to trivially find such a trigger point. Moreover, that information is independent of $g(R, m^*)$, as R is chosen after $\mathsf{Eval}_{\text{IMPL}}$ was created. On the other hand, since $|\sigma^*| = |g(R, m^*)|$, there are at most $n - \omega(\lambda)$ bits left in σ^* that can contain information about $g(R, m^*)$. While $g(R, m^*)$ is a uniform value in the range of $\mathsf{Eval}_{\text{SPEC}}$, it follows that for any σ^*, $\Pr[\mathsf{Eval}_{\text{IMPL}}(\sigma^*) = g(R, m^*)] = \mathsf{negl}(\lambda)$. □

G_4 is essentially the original FDH security game, thus putting together all those lemmas, we can complete the proof. □

References

1. M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *Journal of Cryptology*, 29(4):833–878, Oct. 2016.
2. G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 364–375. ACM Press, Oct. 2015.
3. M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, Apr. 2015.
4. M. Bellare, J. Jaeger, and D. Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1431–1440. ACM Press, Oct. 2015.
5. M. Bellare, K. G. Paterson, and P. Rogaway. Security of symmetric encryption against mass surveillance. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, Aug. 2014.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
7. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.
8. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr. 2008.
9. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
10. R. Chen, Y. Mu, G. Yang, W. Susilo, F. Guo, and M. Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 844–876. Springer, Heidelberg, Dec. 2016.
11. J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, Aug. 2000.
12. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, Aug. 2005.
13. J. P. Degabriele, P. Farshim, and B. Poettering. A more cautious approach to security against mass surveillance. In G. Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 579–598. Springer, Heidelberg, Mar. 2015.
14. Y. Desmedt. Abuses in cryptography and how to fight them. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 375–389. Springer, Heidelberg, Aug. 1990.
15. Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, and T. Ristenpart. A formal treatment of backdoored pseudorandom generators. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, Apr. 2015.

16. Y. Dodis, I. Mironov, and N. Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372. Springer, Heidelberg, Aug. 2016.
17. M. Fischlin and S. Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 76–90, 2018.
18. F. Giacon, F. Heuer, and B. Poettering. KEM combiners. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218. Springer, Heidelberg, Mar. 2018.
19. C. Liu, R. Chen, Y. Wang, and Y. Wang. Asymmetric subversion attacks on signature schemes. In W. Susilo and G. Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 376–395. Springer, Heidelberg, July 2018.
20. U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, Feb. 2004.
21. I. Mironov and N. Stephens-Davidowitz. Cryptographic reverse firewalls. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, Apr. 2015.
22. N. Perlroth, J. Larson, and S. Shane. NSA able to foil basic safeguards of privacy on web. The New York Times, September 2013.
23. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Cliptography: Clipping the power of kleptographic attacks. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, Dec. 2016.
24. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Destroying steganography via amalgamation: Kleptographically CPA secure public key encryption. Cryptology ePrint Archive, Report 2016/530, 2016. http://eprint.iacr.org/2016/530.
25. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Generic semantic security against a kleptographic adversary. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 17*, pages 907–922. ACM Press, Oct. / Nov. 2017.
26. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Correcting subverted random oracles. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 241–271. Springer, Heidelberg, Aug. 2018.
27. A. Young and M. Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, Aug. 1996.
28. A. Young and M. Yung. Kleptography: Using cryptography against cryptography. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.
29. C. Zhang, D. Cash, X. Wang, X. Yu, and S. S. M. Chow. Combiners for chosen-ciphertext security. In T. N. Dinh and M. T. Thai, editors, *COCOON 2016*, volume 9797 of *LNCS*, pages 257–268. Springer, Heidelberg, Aug. 2016.

A The Model: Crooked Indifferentiability

A.1 Preliminary: Indifferentiability

The notion of indifferentiability proposed in the elegant work of Maurer *et al.* [20] has been found very useful for studying the security of hash function and many

other primitives. This notion is an extension of the classical notion of indistinguishability, when one or more oracles are publicly available. The indifferentiability notion is originally given in the framework of random systems [20] providing interfaces to other systems. Coron *et al.* [12] demonstrate an equivalent indifferentiability notion but in the framework of Interactive Turing Machines (as in [9]). The indifferentiability formulation in this subsection is essentially taken from Coron *et al.* [12]. In the next subsection, we will introduce our new notion, *crooked indifferentiability*.

Defining indifferentiability. An ideal primitive is an algorithmic entity which receives inputs from one of the parties and returns its output immediately to the querying party. We now proceed to the definition of indifferentiability [20,12]:

Definition 3 (Indifferentiability [20,12]). *A Turing machine C with oracle accesses to an ideal primitive $\mathcal{G}$ is said to be $(t_{\mathcal{D}}, t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from an ideal primitive $\mathcal{F}$, if there is a simulator $\mathcal{S}$, such that for any distinguisher $\mathcal{D}$, it holds that:*

$$\left|\Pr[\mathcal{D}^{C,\mathcal{G}} = 1] - \Pr[\mathcal{D}^{\mathcal{F},\mathcal{S}} = 1]\right| \leq \epsilon\,.$$

The simulator $\mathcal{S}$ has oracle accesses to $\mathcal{F}$ and runs in time at most $t_{\mathcal{S}}$. The distinguisher $\mathcal{D}$ runs in time at most $t_{\mathcal{D}}$ and makes at most q queries. Similarly, $C^{\mathcal{G}}$ is said to be (computationally) indifferentiable from $\mathcal{F}$ if ϵ is a negligible function of the security parameter λ (for polynomially bounded $t_{\mathcal{D}}$ and $t_{\mathcal{S}}$). See Figure 10.

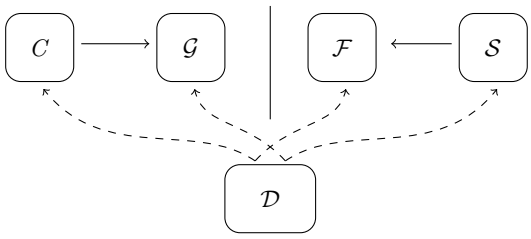

Fig. 10. Indifferentiability: Distinguisher $\mathcal{D}$ either interacts with algorithm C and ideal primitive $\mathcal{G}$, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$. Algorithm C has oracle access to $\mathcal{G}$, while simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$.

As illustrated in Figure 10, the role of the simulator is to simulate the ideal primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with C and $\mathcal{G}$, or with $\mathcal{F}$ and $\mathcal{S}$; in other words, the output of $\mathcal{S}$ should look "consistent" with what the distinguisher can obtain from $\mathcal{F}$. Note that the simulator does not see the distinguisher's queries to $\mathcal{F}$; however, it can call $\mathcal{F}$ directly when needed for the simulation.

Replacement. It is shown that [20] if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem is at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model.

We use the definition of [20] to specify what it means for a cryptosystem to be at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. A cryptosystem is modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an environment $\mathcal{E}$ which provides a binary output and also runs the adversary. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle access to $\mathcal{F}$. The definition is illustrated in Figure 11.

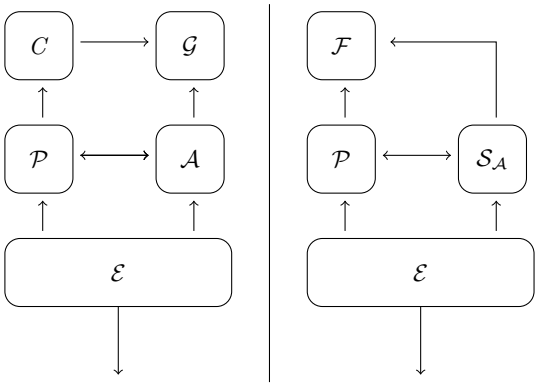

Fig. 11. Environment $\mathcal{E}$ interacts with Cryptosystem $\mathcal{P}$ and Attacker $\mathcal{A}$: In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle access to $\mathcal{F}$.

Definition 4. *A cryptosystem is said to be at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$ model, there exists an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\mathcal{E}(\mathcal{P}^{C}, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\mathcal{E}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \leq \epsilon.$$

where ϵ is a negligible function of the security parameter λ. Similarly, a cryptosystem is said to be computationally at least as secure, etc., if $\mathcal{E}$, $\mathcal{A}$, and $\mathcal{S}_{\mathcal{A}}$ are polynomial-time in λ.

We have the following security-preserving (replacement) theorem, which says that when an ideal primitive is replaced by an indifferentiable one, the security of the "bigger" cryptosystem remains.

Theorem 3 ([20,12]). *Let $\mathcal{P}$ be a cryptosystem with oracle accesses to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model.*

A.2 Crooked Indifferentiability

The ideal primitives that we focus on in this paper are random oracles. A random oracle [6] is an ideal primitive which provides a random output for each new query, and for the identical input queries the same answer will be given. Next, we will formalize a new notion called crooked indifferentiability. Our formalization is for random oracles. We remark that the formalization can be trivially extended for all ideal primitives.

Crooked indifferentiability for random oracles. As mentioned in the Introduction, we are considering to repair a subverted random oracle, such that the corrected construction can be used as good as an unsubverted one. It is thus natural to consider the indifferentiability notion. However, we need to adjust the notion to reflect the subversion and to avoid trivial impossibility. There are two main modifications to the original indifferentiability notion.

1. The deterministic construction will have oracle accesses to the random oracle only via the subverted implementation $\tilde{H}$ but not via the ideal primitive H. This creates lots of difficulty (and even impossibility) for us to develop a suitable construction. For that reason, the construction is allowed to access to trusted but public randomness r.
2. The simulator will also have oracle accesses to the subverted implementation $\tilde{H}$ and also the public randomness r.

The second one is necessary. It is clearly impossible to have an indifferentiability definition with a simulator that has no accesses to $\tilde{H}$, as the distinguisher can simply make query an input such that C will use a value that is modified by $\tilde{H}$ while $\mathcal{S}$ has no way to reproduce it. More importantly, we will show below that, the security will still be preserved to replace an ideal random oracle with a construction satisfying our definition (with an augmented simulator). We will prove the security-preserving (*i.e.*, replacement) theorem from [20] and [12] similarly with our adapted notions.

Definition 5 (H-crooked indifferentiability). *Consider a distinguisher $\widehat{\mathcal{D}}$ and the following multi-phase real execution.*

Initially, the distinguisher $\widehat{\mathcal{D}}$ who has oracle accesses to ideal primitive H, publishes a subverted implementation of H, and denotes it by $\tilde{H}$.

Secondly, a uniformly random string r is sampled and published.

Thirdly, a deterministic construction C is developed: the construction C has random string r as input, and has oracle accesses to $\tilde{H}$ (which can be considered as a crooked version of H).

Finally, the distinguisher $\widehat{\mathcal{D}}$, after having random string r as input, and oracle accesses to the pair (C, H), returns a decision bit b. Often, we call $\widehat{\mathcal{D}}$ the H-crooked-distinguisher.

In addition, consider the corresponding multi-phase ideal execution with the same H-crooked-distinguisher $\widehat{\mathcal{D}}$, where ideal primitive $\mathcal{F}$ is provided.

The first two phases are the same (as those in the real execution).

In the third phase, a simulator $\mathcal{S}$ will be developed: the simulator has random string r as input, and has oracle accesses to $\tilde{H}$, as well as the ideal primitive $\mathcal{F}$.

In the last phase, the H-crooked-distinguisher $\widehat{\mathcal{D}}$, after having random string r as input, and having oracle accesses to an alternative pair $(\mathcal{F}, \mathcal{S})$, returns a decision bit b.

We say that construction C is $(t_{\widehat{\mathcal{D}}}, t_{\mathcal{S}}, q, \epsilon)$-$H$-crooked-indifferentiable from ideal primitive $\mathcal{F}$, if there is a simulator $\mathcal{S}$ so that for any H-crooked-distinguisher $\widehat{\mathcal{D}}$, it satisfies that the real execution and the ideal execution are indistinguishable. Specifically, the following difference should be upper bounded by $\epsilon(\lambda)$:

$$\left| \Pr_{u,r,H}\left[\tilde{H} \leftarrow \widehat{\mathcal{D}} \ : \ \widehat{\mathcal{D}}^{C^{\tilde{H}}(r),H}(\lambda, r) = 1\right] - \Pr_{u,r,\mathcal{F}}\left[\tilde{H} \leftarrow \widehat{\mathcal{D}} \ : \ \widehat{\mathcal{D}}^{\mathcal{F},\mathcal{S}^{\tilde{H},\mathcal{F}}(r)}(\lambda, r) = 1\right] \right|.$$

Here u is the coins of $\widehat{\mathcal{D}}$, $H : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ and $\mathcal{F} : \{0,1\}^{k} \to \{0,1\}^{k}$ denote random functions. See Figure 12 for a detailed illustration of the last phase in both the real and ideal executions.

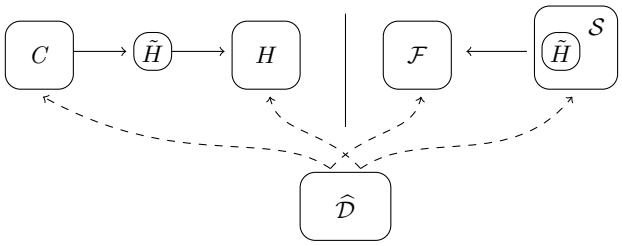

Fig. 12. H-crooked Indifferentiability: distinguisher $\widehat{\mathcal{D}}$, in the first phase, manufactures and publishes a subverted implementation denoted by $\tilde{H}$, for ideal primitive H; then in the second phase, a random string r is published; after that, in the third phase, algorithm C, and simulator $\mathcal{S}$ are developed; the H-crooked-distinguisher $\widehat{\mathcal{D}}$, in the last phase, either interacting with algorithm C and ideal primitive H, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$, returns a decision bit. Here, algorithm C has oracle accesses to $\tilde{H}$, while simulator $\mathcal{S}$ has oracle accesses to $\mathcal{F}$ and $\tilde{H}$.

B Stego-Free Specifications for Randomness Generation and Randomized Algorithms with Known Input Distribution

We recall the definition of stego-free randomness generation and stego-free randomized algorithms with public input distributions [25], and the general results that yield stego-free specifications for them in the trusted-amalgamation model.

Definition 6 (Stego-free randomness generation [25, Definition 3.1]). *For a randomized algorithm* G *with specification* G_{SPEC}*, we say such specification* 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 game in Figure 13 with challenger* $\mathcal{C}$*, at least one of the following conditions hold:*

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

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

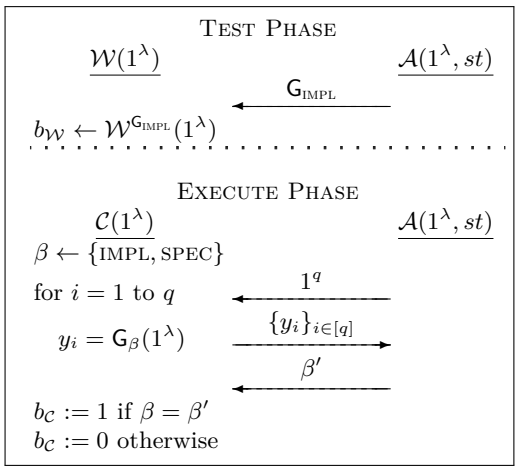

Fig. 13. Stego-Freeness Game for Randomness Generation

Theorem 4 ([25, Theorem 3.4]). *Consider randomness generation* RG *with specification* $(\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi_{\text{SPEC}})$ *as described below (see Figure 14):*

- *Given* 1^λ*,* $\mathsf{RG}^0_{\text{SPEC}}$ *and* $\mathsf{RG}^1_{\text{SPEC}}$ *output uniformly random strings of length* λ*;*
- Φ_{SPEC} *is a hash function so that* $\Phi_{\text{SPEC}}(w)$ *has length* $\lceil |w|/2 \rceil$*;*
- *the specification for* $\mathsf{RG}(1^\lambda)$ *is the trusted composition:* $\Phi_{\text{SPEC}}(\mathsf{RG}^0_{\text{SPEC}}(1^\lambda), \mathsf{RG}^1_{\text{SPEC}}(1^\lambda))$*.*

Then $\mathsf{RG}_{\text{SPEC}}$ *is stego-free if* Φ_{SPEC} *is modeled as a random oracle.*

Note that the above theorem only asserts how to purify randomness generation algorithm G in the random oracle model by splitting G into a constant number of components. It is possible to extend the result to the standard model if we are willing to have polynomially many segments. Such result is demonstrated in the full version [24] of [25]. We quote their result as follows:

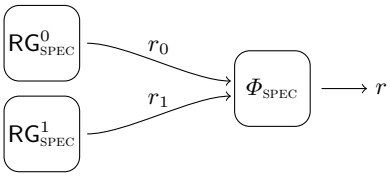

Fig. 14. Subversion-Resistant Specification for Randomness Generation

Proposition 1 ([24]). *There exists a specification for the randomness generation that outputs n bits that is stego-free with the trusted amalgamation and $O(n^\epsilon / \log n)$ segments for any constant ϵ. Similar results hold for randomized algorithms with public input distribution.*

The definition and theorems above cover elementary randomness generation algorithms that only takes a security parameter as input. They can be generalized to consider algorithms that take additional inputs from a large domain in which the adversary specifies a randomized input generator IG, which implicitly defines $\mathsf{G}(1^\lambda, \mathsf{IG}(1^\lambda))$. This class of randomized algorithm includes key generation and bit encryption *etc.*

Formally, let G be a randomized algorithm using λ random bits for inputs of length n. The stego-free game is revised as follows: the challenges $\{y_i\}$ 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. The watchdog is provided oracle access to IG to test G_{IMPL}.

Definition 7 (Stego-free randomized algorithm [25, Definition 3.2]). *For a randomized algorithm G, we say the specification G_{SPEC} is stego-free in the offline-watchdog model, if there exists an offline* PPT *watchdog $\mathcal{W}$, for any* PPT *adversary $\mathcal{A}$ playing the following game in Figure 15 with challenger $\mathcal{C}$, such that either*

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

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

Russell *et al.* [25] established a general transformation yielding a stego-free specification for randomized algorithms with a public input distribution. Consider a randomized algorithm G which uses λ random bits for inputs of length n. Let $(\mathsf{dG}, \mathsf{RG})$ denote the natural specification of G that isolates randomness generation: $\mathsf{RG}(1^\lambda)$ produces λ uniformly random bits and $\mathsf{dG}(r, m)$ is a deterministic algorithm so that for every $m \leftarrow \mathsf{IG}(1^\lambda)$, $\mathsf{G}(m)$ is equal to $\mathsf{dG}(\mathsf{RG}(1^\lambda, m))$ for $n = |m|$. Consider the transformed specification for G of the form $(\mathsf{RG}_0, \mathsf{RG}_1, \Phi, \mathsf{dG})$ where dG is as above. $\mathsf{RG}_0(1^\lambda)$ and $\mathsf{RG}_1(1^\lambda)$ output λ uniform bits, and Φ is a hash function that carries strings of length 2λ to strings of length λ. We have the following theorem:

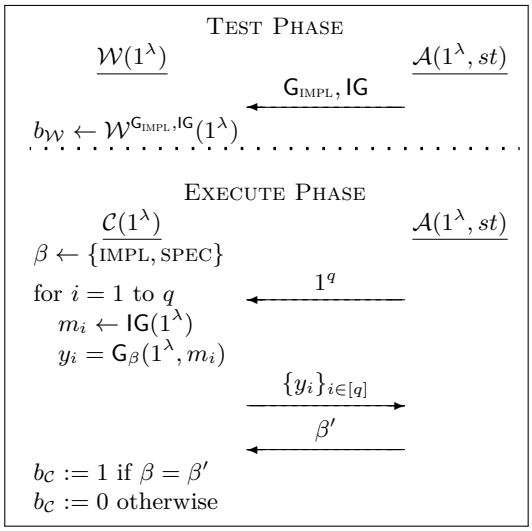

Fig. 15. Stego-Freeness Game for Randomized Algorithms with Input Distribution $\{1^\lambda\} \times \mathsf{IG}$

Theorem 5 ([25, Theorem 3.5]). *For any randomized algorithm* G*, consider the specification* $\mathsf{G}_{\text{SPEC}} := (\mathsf{RG}_{\text{SPEC}}, \mathsf{dG}_{\text{SPEC}})$*, where* $\mathsf{RG}_{\text{SPEC}}$ *and* $\mathsf{dG}_{\text{SPEC}}$ *are as above. Let* $(\mathsf{RG}^0_{\text{SPEC}}, \mathsf{RG}^1_{\text{SPEC}}, \Phi_{\text{SPEC}})$ *be the double-split specification of* $\mathsf{RG}_{\text{SPEC}}$ *as in Figure 14.* G_{SPEC} *is stego-free with a trusted amalgamation (according to Definition 7). Here* Φ_{SPEC} *is modeled as a random oracle.*

C Signature Schemes

A signature scheme is a triple of algorithms $\mathcal{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$. The KGen algorithm takes as input the security parameter λ and outputs a pair of verification/signing key $(\mathsf{vk}, \mathsf{sk})$. The Sign algorithm takes as input sk, a message $m \in \mathcal{M}$ (and random coins $r \in \mathcal{R}$ if Sign is probabilistic), and outputs a signature $\sigma \in \Sigma$. The Verify algorithm takes as input vk and a pair (m, σ) and outputs a bit indicating whether the signature is valid for message m under vk.

Definition 8 (Existential unforgeability). *Let* $\mathcal{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme. We say that* $\mathcal{SS}$ *is* (t, q, ϵ)*-existentially unforgeable under adaptive chosen-message attack (*$\mathsf{EUF\text{-}CMA}$*-secure) if for all* PPT *adversaries* $\mathcal{A}$ *running in time* t *it holds:*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1 & (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda); \\ \wedge\ m^* \notin \mathcal{Q} & (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk}) \end{array}\right] \leq \epsilon$$

where $\mathcal{Q} = \{m_1, \ldots, m_q\}$ *denotes the set of queries to the signing oracle. Whenever* $\epsilon(\lambda) = \mathsf{negl}$ *and* $q = \mathsf{poly}$*, we simply say that* $\mathcal{SS}$ *is* $\mathsf{EUF\text{-}CMA}$*-secure.*