# On the Instantiability of Hash-and-Sign RSA Signatures

Yevgeniy Dodis[1], Iftach Haitner[2,⋆], and Aris Tentes[1]

[1] Department of Computer Science, New York University
{dodis,tentes}@cs.nyu.edu
[2] School of Computer Science, Tel Aviv University
iftachh@cs.tau.ac.il

**Abstract.** The hash-and-sign RSA signature is one of the most elegant and well known signatures schemes, extensively used in a wide variety of cryptographic applications. Unfortunately, the only existing analysis of this popular signature scheme is in the random oracle model, where the resulting idealized signature is known as the RSA *Full Domain Hash* signature scheme (RSA-FDH). In fact, prior work has shown several "uninstantiability" results for various abstractions of RSA-FDH, where the RSA function was replaced by a family of trapdoor random permutations, or the hash function instantiating the random oracle could not be keyed. These abstractions, however, do not allow the reduction and the hash function instantiation to use the algebraic properties of RSA function, such as the multiplicative group structure of $\mathbb{Z}_n^*$. In contrast, the multiplicative property of the RSA function is critically used in many standard model analyses of various RSA-based schemes.

Motivated by closing this gap, we consider the setting where the RSA function representation is generic (i.e., black-box) *but multiplicative*, whereas the hash function itself is in the standard model, and can be keyed and exploit the multiplicative properties of the RSA function. This setting abstracts all known techniques for designing provably secure RSA-based signatures in the standard model, and aims to address the main limitations of prior uninstantiability results. Unfortunately, we show that it is still impossible to reduce the security of RSA-FDH to any natural assumption even in our model. Thus, our result suggests that in order to prove the security of a given instantiation of RSA-FDH, one should use a non-black box security proof, or use specific properties of the RSA group that are not captured by its multiplicative structure alone. We complement our negative result with a positive result, showing that the RSA-FDH signatures *can* be proven secure under the *standard* RSA assumption, provided that the number of signing queries is *a-priori bounded*.

**Keywords:** RSA Signature, Hash-and-Sign, Full Domain Hash, Random Oracle Heuristic, Generic Groups, Black-Box Reductions.

⋆ Research supported by Check Point Institute for Information Security, and ISF grant 1076/11.

R. Cramer (Ed.): TCC 2012, LNCS 7194, pp. 112–132, 2012.

## 1 Introduction

Bellare and Rogaway, [3], introduced the *random oracle* (RO) model, as a "paradigm for designing efficient protocols". When following this paradigm, one first builds a provably secure scheme assuming that an access to a *random* function is given, and (possibly) assuming some "standard" hardness assumption (e.g., factoring is hard). Then it instantiates the scheme by replacing the random function with some *concrete* "hash function" (e.g., SHA-1). The intuition underlying this paradigm is that a successful attack on the resulting scheme should indicate (unexpected) weaknesses of the hash function used. This paradigm (also known as the random oracle heuristic) has led to several highly efficient and widely used in practice constructions, such as the RSA *Full Domain Hash* signature scheme (RSA-FDH) [3] and RSA *Optimal Asymmetric Encryption Padding* scheme (RSA-OAEP) [4]. Typically, however, little is known about the provable security of such popular schemes in the *standard model*. In particular, it is unknown whether we can reduce their security to some "natural" assumption.

In this work we revisit this question once again, focusing, in particular, on the instantiability of the RSA hash-and-sign signatures. The RSA signature [31] is one of the most elegant and well known signatures schemes. It is extensively used in a wide variety of applications, and serves as the basis of several existing standards such as PKCS #1 [32]. In its "textbook" form, the signature $\sigma$ of the message $m$ is simply $\sigma = m^d \bmod n$, which can be verified by checking if $\sigma^e \equiv m \bmod n$, where $e$ is the public RSA exponent, and $d = e^{-1} \bmod \phi(n)$. Of course, the textbook variant is completely insecure, as any $\sigma$ is a valid signature of some message $m = \sigma^e \bmod n$. The traditional fix, known as RSA *hash-and-sign* signature, is to hash the message $m$ before signing it using some "appropriate" hash function $h$ (i.e., $\sigma = h(m)^d \bmod n$). The key question is how to instantiate this function $h$?

Bellare and Rogaway, [3], showed that in the random oracle model, where $h$ is modeled as a truly random function (freely available to all the parties including the adversary), the resulting RSA hash-and-sign signature (which they called RSA *Full Domain Hash*, for short, RSA-FDH) is secure assuming that the (standard) RSA assumption holds. When considering an actual instantiation of $h$, though, a moment's reflection shows that all known security notions for hash functions, such as collision-resistance or pseudorandomness, do not appear to help. In fact, even more "esoteric" notions, such as perfect one-way hash functions or verifiable random functions [5], are not sufficient either. On the other hand, no significant attacks on RSA-FDH signatures are known when $h$ is instantiated using popular "cryptographic hash functions", such as SHA-1. This gave rise to the following important question, which is the main focus of this paper.

> *Is there an instantiation of* RSA-FDH *signature scheme (namely, of the hash function $h$) that can be proven secure under a natural assumption in the standard model?*

Of course, for any concrete hash function, one can "reduce" the security of RSA-FDH signatures to that of RSA-FDH signatures, which is not very useful. So it is important that the assumption used to argue the security of the scheme should be considerably simpler than the chosen message attack on RSA signatures. The best case scenario would be a reduction to the one-wayness of the RSA function (i.e., the standard "RSA assumption"), which is indeed what happened in the idealistic RO model. Unfortunately, we seem to be very far from this goal. In fact, several works, which we survey next, showed various arguments suggesting that no such reduction is likely to exist.

**Existing Impossibility Results.** It is well known that in the general case the random oracle heuristic is false. Specifically, there exist schemes secure in the random oracle model that cannot be instantiated by any concrete hash function [8,7,26,18,2]. Most counter-examples of this kind, however, are rather artificial, and do not shed much light on the security of concrete schemes used in practice. The work that seems most relevant to the focus of this paper is those of [12] and [27] described below (whereas other related work is discussed in Section 1.3).

Dodis et.al., [12], considered a generalization of RSA-FDH signatures, known as (general) *Full Domain Hash* (FDH) signatures. In such signatures, the signer has access to an arbitrary trapdoor permutation $f$, and sets $\sigma = f^{-1}(h(m))$.[1] The main result of [12] rules out proving the security of an instantiation FDH, by reducing it to the one-wayness of $f$ (or more generally, to any assumption on $f$ that is satisfied by a random trapdoor permutation). Their result, however, does not capture reductions that use additional assumptions about $f$. In particular, it seems likely that if a proof of security of some instantiation of RSA-FDH does exist, then it would use the *algebraic properties* of the RSA function. To demonstrate this point, we present (see Section 1.1) an instantiation of RSA-FDH under the standard RSA assumption, that is secure as long as the number of signing queries is a-priori bounded.[2] Our reduction is black box, and critically uses the algebraic properties of $\mathbb{Z}_n^*$. (Indeed, [12] showed that even *one-time* security of *general* FDH signatures cannot be black-box reduced to the one-wayness of the trapdoor permutation.) In addition, the "RSA-based" signatures [16,10,22], which can be proven secure in the standard model (but, alas, no longer have the simple syntax of the RSA signature), critically use the algebraic properties of the RSA function. Finally, even in the random oracle model, tighter security bounds are sometimes achieved using the algebraic properties of RSA (cf., [9], as compared to the generic proofs from trapdoor permutations [3,13]).

More recently, Paillier, [27], looked at the question of instantiating RSA-FDH using a *fixed* hash function (as opposed to a keyed family), and showed that no such instantiation can be black-box reduced to the traditional RSA assumption,

[1] As in the case of RSA-FDH signatures, FDH signatures are known to be secure when the hash function is modeled as a truly random function [3].

[2] With a different motivation, the same result was independently obtained by [21].

assuming the so called "RSA non-malleability" assumption. Informally, this assumption states that calling the RSA inverter on arbitrary "permitted" inputs $(n', e') \neq (n, e)$ does not help in breaking the instance $(n, e)$. We remark that, as observed by Paillier in [27], this assumption is false for various reasonable interpretations of "permitted" tuples $(n', e')$. More significantly, although the restriction to a fixed hash function $h$ is consistent with the existing use in practice, from a theoretical perspective this assumption is somewhat restrictive. For example, while the result of [27] rules out proving even *one-time* security of RSA-FDH, our positive result (see Section 1.1) circumvents this impossibility result by using a keyed hash family.

### 1.1 Our Results

Our main result is a new negative result regarding the instantiability of RSA-FDH, which addresses some of the limitations of the previous negative results of [12,27]. To motivate this result, we start by describing our already mentioned positive result.

**Theorem 1 (Informal).** *Under the standard* RSA *assumption, for every polynomial $t$ there exists an instantiation of* RSA-FDH *that is existentially unforgeable against $t(k)$ signing queries (where $k$ is the security parameter). Furthermore, the reduction treats the group $\mathbb{Z}_n^*$ and the potential adversary in a black-box way.*

The claimed construction is fully described in the full version [], but here we highlight some of its features. First, the result on works for bounded values of $t$, since the constructed hash function description length, is polynomial (quadratic) in the number of signing queries. Second, our construction uses a keyed family of hash functions (which is needed to overcome the impossibility result of [27]). Third, the hash function depends on the RSA modulus $n$ and critically uses the multiplicative structure of the RSA function (which is needed to overcome one of the impossibility result of [12]). Finally, our reduction does not use any other properties of the RSA function besides its multiplicative homomorphism over $\mathbb{Z}_n^*$. Formally, this means that the reduction works given only oracle access to the multiplication and the inversion operations of $\mathbb{Z}_n^*$.

We now turn to our main, negative result, which can be informally stated as follows:

**Theorem 2 (Informal).** *It is impossible to reduce the security of an instantiation of* RSA-FDH *to a "natural" assumption (and in particular to the hardness of* RSA*), provided that (1) the reduction treats the potential adversary in a black-box way; (2) the public exponent $e$ used by the scheme is prime with non-negligible probability; (3) the instantiation only "uses the multiplicative properties of $\mathbb{Z}_n^*$", and should "relativize" to any group isomorphic to $\mathbb{Z}_n^*$.*

We now explain this result in more detail. First, our result holds even if the hash function $h$ is allowed to be keyed, and, moreover, to depend on the RSA modulus $n$ (which was used in our positive result). More significantly, we allow both the

hash function and the hypothetical security reduction $\mathsf{R}$ to use the multiplicative structure of $\mathbb{Z}_n^*$. Finally, we not only rule out reductions to the standard RSA assumption, but also to other non-interactive "RSA-type" assumptions, such as the "strong RSA assumption".

However, our result also has three limitations, (1)-(3). First, and least important, is the assumption that the reduction must treat the adversary in a black-box way. This limitation is met by most existing reductions, and also quite standard in most black-box impossibility results. Technically, it means that the reduction should work given oracle access to any (even inefficient) attacker breaking the security of RSA-FDH. Second, and more significant, is the fact that our current proof relies on the fact that the instantiation will use a prime exponent $e$ (at least with non-negligible probability). Although this limitation appears to be an odd artifact of our specific proof technique, and also seems to be met by most known RSA instantiations, it does leave a possibility for a secure RSA-FDH instantiation always using some composite exponent $e$. Finally, and most significantly, we assume that the reduction "treats the multiplicative RSA group $\mathbb{Z}_n^*$ in a black-box manner". This is formalized (see Section 2) using the notion of *generic groups* [33,25,23]. Informally, though, it means that nothing is assumed about a group element, apart from what was revealed through the performed group operations (i.e., multiplication, inverse and equality check). In particular, an algorithm that treats $\mathbb{Z}_n^*$ in a black-box way should perform equally well given oracle access to any group isomorphic to $\mathbb{Z}_n^*$ (without knowing the isomorphism).

With this intuition in mind, we can interpret Theorem 2 as an indication that in order to prove the security of a given instantiation of RSA-FDH, one should use a non-black box security proof, or use properties of the RSA group, that are not captured by the generic group abstraction. To the best of our knowledge, *all* known positive results on building "RSA-type" signatures — including our new positive result in Theorem 1, the standard model constructions of [16,10,22], and the random-oracle based analysis of [3,9] — treat $\mathbb{Z}_n^*$ as a black-box, and only use its multiplicative structure. Thus, although still restrictive, our result rules out all known techniques for proving the security of RSA-based signatures, which was not the case for the previous results of [12,27]. Still, the restriction of the reduction to only use the multiplicative structure of $\mathbb{Z}_n^*$ is quite significant, which raises the question if this restriction could be relaxed.

**Removing Generic Groups?** Unfortunately, removing (or even relaxing) the above mentioned restriction appears to be very challenging. Intuitively, with our current techniques (see more below) we must be able to construct an algorithm $\mathsf{Forger}$ which, given any (family of) hash function(s) $h$, should be able to (1) break the RSA-FDH instantiation using this $h$, and, yet, (2) do so by only forging the signature which the reduction $\mathsf{R}$ must already "know" (so that $\mathsf{Forger}$ never helps $\mathsf{R}$ compute something which $\mathsf{R}$ does not know to begin with, potentially helping $\mathsf{R}$ to break some hardness assumption). In particular, satisfying conflicting properties (1) and (2) seems to require some kind of "reverse-engineering" (or "de-obfuscation") techniques on $h$ which seem to be completely beyond our current capabilities, without placing any restriction on the reductions we allow.

Indeed, the introduction of the generic group model was precisely the step which (a) allowed our forger to "reverse engineer" the given hash function $h$ (so as to provably satisfy properties (1)-(2) above), and, yet, (b) allowed the reduction to use the algebraic properties of $\mathbb{Z}_n^*$.

### 1.2 Our Technique

On a very high level, our proof follows the approach of [12] used to prove that there exists no fully black-box reduction from (general) FDH signature schemes to the one-wayness of random functions. [12] defined an oracle Forger relative to which no FDH signature scheme is secure, yet Forger does not help inverting a random function. In more detail, on input $(h, \{\sigma_i\}_{i\in[t]})$, Forger checks that (1) $\{\sigma_i\}$ are valid signatures for the messages $1, \ldots, t$ (i.e., $f(\sigma_i) = h(i)$ for every $i \in [t]$, where $f$ is the random function), (2) the evaluation of $h(1), \ldots, h(t)$ does not query $f$ on any element of $\{\sigma_i\}$, and (3) $t$ is at least equal to $|h|$ – the description size of $h$. If positive, Forger returns the signature of 0 (i.e., $f^{-1}(h(0))$).

It is clear that Forger can be used to break the existential security of any FDH scheme: the attacker uses Sign, the signer of the scheme, to compute $\{\sigma_i\}_{i\in[t]}$ for some $t \geq |h|$, and then calls Forger on $(h, \{\sigma_i\})$, where we assume without loss of generality that condition (2) above holds with respect to this query (otherwise, faking a signature *without* Forger is easy). On the other hand, [12] showed that an efficient algorithm (with oracle access to $f$, but not to Sign) cannot provide all these signatures. Thus, Forger is useless in these settings, and in particular a black-box reduction (i.e., algorithm) cannot make use of Forger for inverting a random function, proving the main result of [12].

Intuitively, Forger is useless for an algorithm with no access to Sign, for the following reason. Fix some efficient oracle-aided algorithm R and let $\{0,1\}^n$ be the domain of the random function $f$. Since a random function is one way, the only elements that R can invert are those elements it previously received as answers to its $f$-queries. Hence (since $f$ is random), R only knows how to invert *random* elements inside $\{0,1\}^n$. Since it takes at least $t$ bits to describe $t$ random elements in $\{0,1\}^n$ (actually, it takes $tn$ bits) and since the evaluation of $h(1), \ldots, h(t)$ does not query $f$ on elements inside $\{\sigma_i\}_{i\in[t]}$, there must exist $h(i) \in \{h(1), \ldots, h(t)\}$ that R does not know how to invert, and thus cannot provide a valid signature for the message $i$.

Moving to our setting, we focus for concreteness on fully black-box reductions from RSA-FDH to the hardness of RSA (i.e., such reductions use the multiplicative RSA group $\mathbb{Z}_n^*$ and the adversary in a black-box way). The blackboxness in the RSA group tells us that such a reduction should work with respect to any group isomorphic to $\mathbb{Z}_n^*$. In particular, it should work well with respect to the group $\pi(\mathbb{Z}_n^*)$, obtained by renaming the elements of $\mathbb{Z}_n^*$ according to a random permutation $\pi$ over $\mathbb{Z}_n^*$ (i.e., $a \cdot b$ is defined as $\pi(\pi^{-1}(a) \cdot \pi^{-1}(b) \bmod n)$).

Given the above understanding, the first attempt would be to define Forger analogously to that of [12]. Namely, on input $(n, e, h, \{\sigma_i\}_{i\in[t]})$, Forger checks that (1) $\sigma_i^e \equiv h(i)$ for every $i \in [t]$, (2) the evaluation of $h(1), \ldots, h(t)$ does

not compute $\sigma_i$ for some $i \in [t]$, and (3) $t \geq |h|$. If positive, Forger returns the signature of 0 (i.e., $h(0)^d$, for $d = e^{-1} \bmod \phi(n)$, where all group operations are over the group $\pi(\mathbb{Z}_n^*)$.

We would like to argue that if $\pi$ is chosen at random, then the only way to make a non-aborting query to Forger is via using Sign, the signer of the scheme. It would then follow that Forger is useless for an algorithm R that has no access to Sign (and in particular to a black-box reduction). It turns out, however, that in our settings such R *can* make non aborting calls to Forger. The issue is that unlike in the setting of [12], R can make use of the algebraic structure of $\mathbb{Z}_n^*$ to construct a non-aborting query to Forger. For instance, R can compute $\{j^e\}_{j\in[\ell]}$, and assuming some reasonable mapping $M$ from $[t = \ell^2]$ to $\{j \cdot k\}_{j,k\in[\ell]}$, let $h(i) = M(i)^e \bmod n$ and $\sigma_i = M(i)$. Since the evaluation of $h(1), \ldots, h(t)$ does not query an element of $\{\sigma_i\}_{i\in[t]}$), it follows that $(n, e, h, \{\sigma_i\}_{i\in[t]})$ is a non-aborting query.[3] Alternatively, if R can break the RSA assumption over $\pi(\mathbb{Z}_n^*)$ (say, if it knows the factorization of $n$), then it can set $h(i) = i$ and compute $\sigma_i = h(i)^d$ (using the factorization of $n$ to compute $d$).

Fortunately, we manage to prove that a non-aborting query of R is either "degenerated" (as in the first example) or indicates that R knows the factorization of $n$. To handle the first case, we change Forger to identify and abort on degenerated queries. Where we also show that it is easy to forge a signature with respect to a degenerated $h$ (i.e., $h$ that is part of a degenerated query), even *without* the help of Forger. Namely, we show that there is no secure RSA-FDH scheme relative to the modified Forger. We then show that with respect to this modified Forger, one can efficiently extract the factorization of $n$ from an algorithm that produces a non-aborting query. It follows that for any efficient algorithm R with oracle access to Forger, there exists an efficient algorithm, with no access to Forger, that emulates $\mathsf{R}^{\mathsf{Forger}}$ well. In other words, we prove that Forger is useless for the class of efficient algorithms with no oracle access to Sign.

Proving the above intuition is the main challenge of this work, and we achieve that using a novel adaptation of the Gennaro-Trevisan, [17], short description paradigm, described below, to the generic groups realm.[4]

**The Gennaro-Trevisan [17] Short Description Paradigm and Its Adaption to Generic Groups.** Loosely, [17] shows that an efficient algorithm that inverts a random function too well, can be used to give a too short description for a random function (and thus cannot exist). This elegant approach has turned to be an extremely powerful approach for proving impossibility results in the random functions realm, which typically imply black-box impossibility results for one-way functions/permutations based constructions. While the Gennaro-Trevisan paradigm (from now on, the GT paradigm) has several extensions (e.g., [15,35,19,20,30]), all are given in the random functions realm.

[3] Note that to describe $h$ it suffices to describe the set $\{j^e\}_{j\in[\ell]}$. Thus $|h| \in O(\ell \log n)$, which is smaller than $t$ for large enough $\ell$.

[4] A side benefit of this proof technique, is an alternative proof to the equivalence of RSA and factoring over generic groups, firstly proven by Aggarwal and Maurer, [1] (the latter, however, also proves it over "generic rings").

We would like to apply a similar approach for arguing that an algorithm that makes a non-aborting query to Forger, can be either used to factor $n$, or to "compress" the random permutation $\pi$ (which defines the group $\pi(\mathbb{Z}_n^*)$). Since compressing $\pi$ is impossible, it follows that a non-aborting query of such an algorithm can be used to factor $n$. Hence, such queries can be answered efficiently, yielding the existence of an efficient emulator (without access to Forger) for any efficient algorithm.[5]

Extending the GT paradigm to our settings involves many complications. The main part of the GT paradigm is using the (hypothetical) attacker to reconstruct a random function using (too) short advice. This reconstruction involves emulating the attacker, where the key point is to do this without "wasting information": any bit used to emulate, should give a bit of information about the (random) function. Doing the latter is quite easy for random functions; the answer to any query of the attacker gives the same amount of information about the function (i.e., the info that it maps the query input to the provided output). The only subtlety is that there are repeated queries (which are clearly wasteful), but handling such queries is easy: simply keep track of the query history on the emulation.

In our setting, however, things get much more complicated. To begin with, there might be non-repeating queries whose answers yield very little information about the random group $\pi(\mathbb{Z}_n^*)$ (and therefore about $\pi$). For instance, for some $n$'s there are only four possible answers for the query $a^{\phi(n)/4}$ over $\pi(\mathbb{Z}_n^*)$. Thus, roughly speaking, the answer for this query contains only two bits of information about $\pi$. More generally, it appears that one can create much more intricate examples; e.g., when the answer to the query follows a very complicated distribution, based on the answers given so far.

An even more challenging task is proving the dichotomy that a non-aborting query can either be used to (efficiently) factor $n$, or implies a (too) short description of $\pi$. Handling the above challenges requires an intimate understanding of the algebraic structure of the group $\mathbb{Z}_n^*$, in particular of the set of solutions for linear equations over this group, and critically uses the fact that factoring is solvable in sub-exponential time [11,34].

### 1.3 Other Related Work

We briefly mention other known results concerning the uninstantiability of popular signature and encryption schemes that can be proven secure in the random oracle model. Paillier and Vergnaud, [28], showed that many popular discrete log based signatures (including *ElGamal*, *DSA* and *Schnorr*) *cannot* be reduced to the discrete log assumption in the standard model, using the so called "algebraic" reductions. (Similar results also hold for related GQ signatures under the RSA assumption.) Although technically incomparable to our "generic group" modeling, conceptually such reductions are related to our assumption that the

[5] In addition, since non-aborting queries are *easy* to generate assuming that RSA is easy over $\pi(\mathbb{Z}_n^*)$, the above would immediately yield that RSA is equivalent to factoring over (random) $\pi(\mathbb{Z}_n^*)$, and thus over generic groups.

reduction can only use the multiplicative structure of a given group. Indeed, in both cases the "meta-reduction" can eventually figure out the multiplicative relations used be the reduction $\mathsf{R}$ in its queries to the attacker. The main difference applies in the way the reduction can prepare its queries to the attacker. While the generic group modeling allows the reduction $\mathsf{R}$ to use some "hidden values" related to the assumption that $\mathsf{R}$ is trying to break, "algebraic" reduction do not allow this flexibility. Thus, much of the technical difficulties in the generic group modeling (e.g., extracting the hidden representations computed by the reduction "on the side") are somewhat trivialized when restricted to "algebraic" reductions. Additionally, the results of [28] are specific to reductions from a concrete assumption (e.g., discrete log), and are conditional on another assumption (e.g., "one-more" discrete log). In contrast, our results are unconditional and rule out all starting assumptions, but only in the generic group model.

Finally, in the realm of factoring/RSA-based CCA encryption, Paillier and Villar, [29] and Brown et.al., [6], showed uninstantiability results analogous to already-mentioned RSA signature result of [27].

### Paper Organization

In Section 2 we formally define RSA-FDH and its security in the generic group model and the type of reductions we rule out. Our main result, regarding the impossibility of existentially unforgeable RSA-FDH against unbounded number of signing queries, is proven in Section 3. However, the proof of our main technical lemma using the GT short description paradigm is omitted and can be found in the full version [14].

## 2 RSA-FDH in the Generic Group Model

In the following we first formally define what we mean by generic group model, then extend the standard definitions of RSA-FDH to this model and finally define weakly black-box proofs of security.

### 2.1 The Generic Group Model

There are different ways to interpret what it means to "treat the multiplicative RSA group $\mathbb{Z}_n^*$ in a black-box way" (see Theorem 2). In the *generic algorithm model* due to Maurer, [23], "generic" algorithms do not have a direct access to the group elements, but rather to a "black box" containing each element. The only operations allowed with these boxes, are the group operations (inverse and multiplication) and comparing two boxes for equality. The formulation we have chosen here, which we simply call the generic group model, is somewhat less abstract. An algorithm in our model has an oracle access to a group isomorphic to

$\mathbb{Z}_n^*$ (specifically, the group resulting by renaming the elements of $\mathbb{Z}_n^*$ according to some random permutation), through which it can perform the group operations. Unlike the generic algorithm model, however, in our model algorithms we do have access to the representation of the group elements and can manipulate them.

Since any algorithm that "works well" in the generic algorithm model (e.g., breaks the RSA assumption) implies an algorithm that works equally well in our model with respect to *any* group isomorphic to $\mathbb{Z}_n^*$, an impossibility result in our model implies a similar result in the model of Maurer, [23]. Namely, our model can be viewed as a model for proving impossibility results in the generic algorithm model.

We formally define our model as follows: for $n \in \mathbb{N}$, let $\Pi_{\phi(n)}$ be the set of all permutations from $\mathbb{Z}_n^*$ to $\mathbb{Z}_n^*$. For $\pi \in \Pi_{\phi(n)}$, we denote with $\pi(\mathbb{Z}_n^*)$ the group induced by the group $\mathbb{Z}_n^*$ where each element of $\mathbb{Z}_n^*$ is renamed according to $\pi$. More specifically, the group operations over $\pi(\mathbb{Z}_n^*)$ are defined as follows: the inverse of $a \in \mathbb{Z}_n^*$ is $\pi((\pi^{-1}(a))^{-1} \bmod n)$ and the (group) product of $a, b \in \pi(\mathbb{Z}_n^*)$ is $\pi(\pi^{-1}(a) \cdot \pi^{-1}(b) \bmod n)$. By $\Pi(\mathbb{Z}_n^*)$ we denote the multiset of all groups $\pi(\mathbb{Z}_n^*)$, where $\mathcal{G} = \{G = \{G_n \colon G_n \in \Pi(\mathbb{Z}_n^*)\}_{n\in\mathbb{N}}\}$ (i.e., $\mathcal{G}$ consists of sets of groups, where each set contains a group of $\Pi(\mathbb{Z}_n^*)$ for every $n \in \mathbb{N}$).

Abusing notation, we view $G \in \mathcal{G}$ as an oracle that given as input $n \in \mathbb{N}$ and one [resp., two elements] of $G_n$ (i.e., of $\mathbb{Z}_n^*$), returns the group inverse [resp., the group product] of the element (if the oracle $G$ is given as input an element outside $G_n$, it returns $\perp$), and let $G_n(\cdot) = G(n, \cdot)$. Given a sequence of group operations (e.g., $a \cdot b^{-1}$), we sometimes add the term $\quad [G_n]$, to indicate that the operations are done with respect to the group $G_n$. In the following, abusing notation again, we will write $G \leftarrow \mathcal{G}$, where this sampling is not well defined because $\mathcal{G}$ is an infinite set. However, we can assume lazy sampling, namely for every query which contains a new $n$, $G_n$ is sampled uniformly at random from $\Pi(\mathbb{Z}_n^*)$ (which is a finite set).

### 2.2 RSA-FDH Signature Schemes in the Generic Group Model

RSA-FDH signature schemes over $G \in \mathcal{G}$ is defined as follows:

**Definition 1 (RSA-FDH signature scheme in the generic group model).** *An* RSA-FDH *signature scheme $\Sigma^{\mathcal{G}}$ in the generic group model, consists of the following triplet of oracle-aided* PPT *'s* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- *Given oracle access to $G \in \mathcal{G}$ and input $1^k$, $\mathsf{KeyGen}^G$ outputs a "public key" $(n, e, h)$, where $n \in \mathbb{N}$ is a product of two primes, $e \in \mathbb{Z}_{\phi(n)}^*$ and $h$ is a (hash) function, represented as an oracle-aided circuit mapping values into $\mathbb{Z}_n^*$, and a "secret key" $d = e^{-1} \bmod \phi(n)$.*
- *Given oracle access to $G \in \mathcal{G}$, input $n \in \mathbb{N}$, $d \in \mathbb{Z}_{\phi(n)}^*$, a circuit $h$ mapping values into $\mathbb{Z}_n^*$ and a "message" $m$ in the domain of $h$, $\mathsf{Sign}^G$ outputs the "signature" $h^G(m)^d \quad [G_n]$.*

– *Given oracle access to $G \in \mathcal{G}$, input $n \in \mathbb{N}$, $e \in \mathbb{Z}^*_{\phi(n)}$, a circuit $h$ mapping values into $\mathbb{Z}^*_n$, a "message" $m$ in the domain of $h$ and $\sigma \in \mathbb{Z}^*_n$,* $\mathsf{Verify}^G$ *outputs one iff $\sigma^e \equiv h^G(m)$ $[G_n]$.*

*For $G \in \mathcal{G}$, we let $\Sigma^G$ be the instantiation of $\Sigma^{\mathcal{G}}$ with $G$.*

**Security Definition.** The following definition realizes the security of bounded and unbounded existential unforgeability under chosen message attack of an RSA-FDH signature in the generic group model, analogously to that of the standard model.

**Definition 2 (security of RSA-FDH signature in the generic group model).** *An oracle-aided algorithm $F$* breaks the security *of an* RSA-FDH *signature scheme $\Sigma^{\mathcal{G}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, if*

$$\mathsf{Pr}_{G \leftarrow \mathcal{G}, (sk,pk) \leftarrow \mathsf{KeyGen}^G(1^k)}[(m, \sigma) \leftarrow F^{G, \mathsf{Sign}^G(sk,pk,\cdot)}(pk) : \mathsf{Verify}^G(\sigma, m, pk) = 1 \wedge \mathsf{Sign} \text{ was not queried on } m] > \mathrm{neg}(k)$$

*A signature scheme $\Sigma^{\mathcal{G}}$ is* EU-CMA*-secure, if no (oracle-aided)* PPT *breaks its security, where $\Sigma^{\mathcal{G}}$ is $t$-*EU-CMA*-secure, if no* PPT *breaks its security when restricted to query* $\mathsf{Sign}$ *at most $t(k)$ times.*

### 2.3 Weakly Black-Box Proofs of Security

Since we would like to rule out an EU-CMA-secure scheme, we ask the security proof of the scheme to be realized via a "black-box reduction" (as discussed in the introduction, we have very little chance to rule out a general proof of security). On the other hand, we consider a very weak form of such a reduction (which strengthens our main impossibility result).

**Definition 3 (weakly black-box proof of security of RSA-FDH).** *An* RSA-FDH *signature scheme $\Sigma^{\mathcal{G}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ in the generic group model has a* weakly black-box proof of security *based on an assumption* $\mathsf{X}$*, if there exists an oracle-aided* PPT *$R$ such that if* $\mathsf{X}$ *is true, then the following holds: let $F$ be a (possibly unbounded) adversary that breaks the security of $\Sigma^{\mathcal{G}}$ (see Definition 2), then for any* PPT $\mathsf{Emul}$ *there exists a polynomial-length distribution ensemble $\mathcal{D} = \{D_k\}_{k \in \mathbb{N}}$ such that*

$$\mathrm{SD}\left((x, R^{G,F^G}(1^k, x)), (x, \mathsf{Emul}^G(1^k, x))\right)_{G \leftarrow \mathcal{G}, x \leftarrow D_k} > \mathrm{neg}(k).^6$$

*Remark 1 (A black-box proof implies a weakly black-box proof).* Assuming that $\mathsf{X}$ is true, the above intuitively asks that a security breach of $\Sigma^{\mathcal{G}}$ implies that a

[6] Note that $F$ is an adversary which expects oracle access to $\mathsf{Sign}$ and $R$ can control the responses of these queries of $F$. The same does not hold for the queries of $F$ to $G$.

(slightly) non-trivial task can be performed. Specifically, an efficient oracle-aided algorithm can use a breaker of the scheme (in a black-box way) to sample some unsamplable distribution. Note that this is a very modest demand and indeed, it is implied by most black-box proofs of security one can think of.

Consider for instance a proof of security $R$ that black-box reduces the security of a scheme $\Sigma^{\mathcal{G}}$ to an assumption $\mathsf{X}$, say to the hardness of factoring. It follows that given any adversary $F$ to $\Sigma^{\mathcal{G}}$, the algorithm $R^{G,F^G}$ factors integers too well. Assume without loss of generality that $R^{G,F^G}(x)$, if succeeds, outputs the factorization of the integer $x$, let $D_k$ be the distribution that outputs an integer $x = pq$, for two randomly chosen $k$-bits prime, and consider the distribution $\xi_k = (x, R^{G,F^G}(1^k, x))_{G\leftarrow\mathcal{G},x\leftarrow D_k}$ it induces. Now if factoring is hard, then there is no efficient $\mathsf{Emul}$ such that $(x, \mathsf{Emul}^G(1^k, x))_{G\leftarrow\mathcal{G},x\leftarrow D_k}$ is (even computationally) close to $\xi_k$. Namely, there is no weakly black-box proof of security for $\Sigma^{\mathcal{G}}$ *based* on factoring.

Now if factoring is hard, then there is no efficient $\mathsf{Emul}$ such that $(x, \mathsf{Emul}^G(1^k, x))_{G\leftarrow\mathcal{G},x\leftarrow D_k}$ is (even computational) close to $\xi_k$. Namely, there is no weakly black-box proof of security for $\Sigma^{\mathcal{G}}$ *based* on factoring.[7]

## 3 There Exists No RSA-FDH with a Weakly Black-Box Proof

In this section we prove the main result of this paper.

**Theorem 3 (Theorem 2, restated).** *Let* $\Sigma^{\mathcal{G}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be an* RSA-FDH *signature scheme in the generic group model in which* $\mathsf{Pr}_{G\leftarrow\mathcal{G},(n,e,h)\leftarrow\mathsf{KeyGen}^G(1^k)}[e \in \mathrm{P}] > \mathrm{neg}(k)$. *If* $\Sigma^{\mathcal{G}}$ *has a weakly black-box proof of security based on (an assumption)* $\mathsf{X}$, *then* $\mathsf{X}$ *is false.*

The proof of Theorem 3 immediately follows from the next lemma:

**Lemma 1.** *Let* $\Sigma^{\mathcal{G}}$ *be as in Theorem 3, then there exist a family of oracles* $\mathsf{Forger} = \{\mathsf{Forger}_G\}_{G\in\mathcal{G}}$ *and oracle-aided* PPT*'s* $F$ *and* $\mathsf{Emul}$, *such that the following hold:*

1. *For every* $G \in \mathcal{G}$, $F^{G,\mathsf{Forger}_G}$ *breaks the security of* $\Sigma^G$.
2. *For any oracle-aided* PPT $A$ *and polynomial-length distribution ensemble* $\mathcal{D} = \{D_k\}_{k\in\mathbb{N}}$:

$$\mathrm{SD}\left((x, A^{G,\mathsf{Forger}_G}(1^k, x)), (x, \mathsf{Emul}^G(1^k, x, \mathrm{desc}(A)))\right)_{G\leftarrow\mathcal{G},x\leftarrow D_k} = \mathrm{neg}(k),$$

*where* $\mathrm{desc}(A)$ *denotes the description of the Turing Machine* $A$.

Before proving Lemma 1, let us first use it for proving Theorem 3.

[7] Note that there nothing specific to the hardness of factoring in the above discussion, but rather it seems to be generic to "any" hardness assumption (e.g., strong RSA).

*Proof (of Theorem 3).* Let $\Sigma^{\mathcal{G}}$ be an RSA-FDH scheme with $\mathsf{Pr}_{G\leftarrow\mathcal{G},(n,e,h)\leftarrow\mathsf{KeyGen}^G(1^k)}[e \in \mathrm{P}] > \mathrm{neg}(k)$. Assume that $\Sigma^{\mathcal{G}}$ has a weakly black-box proof of security based on (an assumption) $\mathsf{X}$ and let $R$ be the algorithm guaranteed by this proof. Let $\mathsf{Emul}$ be the algorithm guaranteed by Lemma 1 with respect to $\Sigma^{\mathcal{G}}$. Lemma 1 yields that

$$\mathrm{SD}\left((x,\tilde{R}^{G,\mathsf{Forger}_G}(1^k,x)),(x,\mathsf{Emul}^G(1^k,x,\mathrm{desc}(\tilde{R}))\right)_{G\leftarrow\mathcal{G},x\leftarrow D_k}=\mathrm{neg}(k)$$

for any polynomial-length distribution ensemble $\mathcal{D} = \{D_k\}$, where $\tilde{R}^{G,\mathsf{Forger}_G}(\cdot) = R^{G,F^{\mathsf{Forger}_G}}(\cdot)$. Letting $\tilde{F}^G(\cdot) = F^{G,\mathsf{Forger}_G}(\cdot)$ and $\mathsf{Emul}^G_R(\cdot) = \mathsf{Emul}^G(\cdot,\mathrm{desc}(\tilde{R}))$, it follows that

$$\mathrm{SD}\left((x,R^{G,\tilde{F}^G}(1^k,x)),(x,\mathsf{Emul}^G_R(1^k,x))\right)_{G\leftarrow\mathcal{G},x\leftarrow D_k}=\mathrm{neg}(k)$$

for any polynomial-length distribution ensemble $\mathcal{D}$, yielding that $\mathsf{X}$ is false.

The rest of this section is devoted for proving Lemma 1. We find it more convenient, however, to prove a variant of Lemma 1 in which the emulator should work for any (polynomial-size) family of circuits. Namely, we prove the following lemma (in the following statement we only focus on the part that changed comparing to the original statement):

**Lemma 2 (non uniform variant of Lemma 1)**

2. *The following holds for any (no input) polynomial-size family of oracle-aided circuits* $\{C_k\}_{k\in\mathbb{N}}$*:*

$$\mathrm{SD}\left(C_k^{G,\mathsf{Forger}_G},\mathsf{Emul}^G(1^k,\mathrm{desc}(C_k))\right)_{G\leftarrow\mathcal{G}}=\mathrm{neg}(k),$$

*where* $C_k^{G,\mathsf{Forger}_G}$ *denotes the output of* $C_k$ *given access to* $G$ *and* $\mathsf{Forger}_G$*, and* $\mathrm{desc}(C_k)$ *denotes the description of* $C_k$*.*

It is easy to see that the non-uniform lemma above yields the uniform Lemma 1. In Section 3.1 we define the family of oracles $\mathsf{Forger}$ and the efficient algorithm $F$ that uses $\mathsf{Forger}$ to break any RSA-FDH scheme, in Section 3.3 we define the emulator $\mathsf{Emul}$, where in Section 3.4 we put things together to prove Lemma 2.

### 3.1 The Forger

Recall (see Section 1.2) that $\mathsf{Forger}$ has to abort on "degenerated queries" — essentially those queries that are easy to produce over any group in $\Pi(\mathbb{Z}_n^*)$. To determine whether a query $(n,e,h,\{\sigma_i\}_{i\in[t]})$ is degenerated, we measure the complexity of the values $\{h(i)\}_{i\in[t]}$,[8] as a function of the group queries done through

[8] We actually mean $\{h^G(i)\}_{i\in[t]}$, but for notational convenience we will sometimes omit the superscript $G$ from $h$.

their evaluations. Since the actual representation of these values is meaningless, we only focus on their representation as functions of the "hardwired terms" — the values used in the evaluation of $\{h(i)\}$ that first appear as an input to a group oracle call. Note that any group element used in the evaluation of $\{h(i)\}$, can be expressed using (only) these hardwired terms. To formally carry the above discussion, we describe the evaluation of $\{h(i)\}$ as a computation over the following group.

**Definition 4 (The group** Symb**).** *The elements of* Symb *are equivalence classes over the set of all finite strings "$u_1^{a_1}, \cdots, u_k^{a_k}$", where the $u_i$'s are in $\mathbb{N}$ and the $a_i$'s are in $\mathbb{Z}$. The strings $c =$ "$u_1^{a_1} \cdot \ldots \cdot u_k^{a_k}$" and $c' =$ "$u_1'^{a_1'} \cdot \ldots \cdot u_{k'}'^{a_{k'}'}$" are in the same equivalence class, if for every $w \in \mathbb{N}$ it holds that $\sum_{i \in [k]:\, u_i = w} a_i = \sum_{i \in [k']:\, u_i' = w} a_i'$. We identify a group element of* Symb*, with any string of its equivalence class. The unit element of* Symb *is the class identified by the empty string $\varepsilon$ (or by "$2^1 \cdot 2^{-1}$" etc), where $c \cdot c'$ is the equivalence class identified by the string "$c \cdot c'$" and finally $c^{-1}$ is the class identified by the string "$u_1^{-a_1} \cdot \ldots \cdot u_k^{-a_k}$".*

We naturally identify an element "$u_1^{a_1} \cdot \ldots \cdot u_k^{a_k}$" $\in$ Symb with an element of a given group $V$ that contains $\{u_i\}_{i \in [k]}$, by identifying it with the result of the sequence of operations it induces over $V$ (i.e., "$u_1 \cdot u_2^{-1}$" with respect to $V = \mathbb{Z}_n^*$, is identified with $u_1 \cdot u_2^{-1} \mod n$). To avoid confusion over which group a sequence of operations is taken, we typically suffix the sequence with the term $[V]$, indicating that it is done over the group $V$. It is clear that for any two strings $u$ and $u'$ that identify the same element of Symb (i.e., belong to the same equivalence class), it holds that $u \equiv u' \ \ [V]$ for any Abelian group $V$ containing $u$ and $u'$.

Next we use the above terminology to syntactically describe the computation of an oracle-aided circuit $C$, where we start by defining the hardwired terms determined by $C$'s computation. To simplify notations, we assume that a circuit evaluates its gates one-by-one, and that its description determines this evaluation order.

**Definition 5 (hardwired terms).** *Let $C$ be an oracle-aided circuit, $G \in \mathcal{G}$ and $n \in \mathbb{N}$. The* terms *of $C$ with respect to $G_n$, denoted* $\text{Terms}_{C,G,n}$*, are those values that appear either as input or as the answers to non-bottom queries of $C$ to $G_n$ (i.e., $G_n$ returns a non-bottom value). The* hardwired terms *of $C$ with respect to $G_n$, denoted* $\text{HardWired}_{C,G,n}$ *are those element inside* $\text{Terms}_{C,G,n}$ *that* first *appear as inputs to non-bottom queries to $G_n$. Finally, the* answer terms *are those terms that appear as answers to non-bottom queries (might intersect* $\text{HardWired}_{C,G,n}$*). We assume that the elements of each of the above sets are ordered according to the evaluation order.*

We next use the syntax of the group Symb, to present any term as an expression of the hardwired terms.

**Definition 6 (canonical form).** *Let $C$, $G$ and $n$ be as in Definition 5. The* canonical form *of $u \in \text{Terms}_{C,G,n}$ with respect to $(C, G, n)$, denoted* $\text{Can}_{C,G,n}(u)$*, is recursively defined as follows:*

- *if* $u \in \text{HardWired}_{C,G,n}$, *let* $\text{Can}_{C,G,n}(u)$ *be the element* "$u^1$" $\in$ Symb.
- *If* $u$ *first appears as an output of a query* $G_n(u', u'')$, *let* $\text{Can}_{C,G,n}(u) = \text{Can}_{C,G,n}(u') \cdot \text{Can}_{C,G,n}(u'')$ [Symb].
- *Similarly, if* $u$ *first appears as an output of* $G_n(u')$, *we let* $\text{Can}_{C,G,n}(u) = \text{Can}_{C,G,n}(u')^{-1}$ [Symb].

Let $\{v_i\}_{i\in[\ell]} = \text{HardWired}_{C,G,n}$. Note that the canonical form of any $u \in \text{Terms}_{C,G,n}$ with respect to $(C, G, n)$, can be *uniquely* written as $\prod_{i\in[\ell]} v_i^{a_i}$ [Symb], where $a_i$ might be non zero, only if the hardwired term $v_i$ appears before $u$ does (in the evaluation order of $C^G$). Finally, the canonical forms of a set of terms, with respect to $(C, G, n)$, is compactly represented using the following matrix.

**Definition 7 (canonical-form matrix).** *Let* $C$, $G$ *and* $n$ *be as in Definition 5, let* $\{v_i\}_{i\in[\ell]} = \text{HardWired}_{C,G,n}$ *and let* $\mathcal{W} = \{u_i\}_{i\in[t]} \subseteq \text{Terms}_{C,G,n}$. *The matrix* $M^{G,n,C}(\mathcal{W}) \in \mathbb{Z}_{t\times\ell}$ *is defined as* $\{a_{ij}\}_{i\in[t],j\in[\ell]}$, *assuming that* $\text{Can}_{C,G,n}(u_i) = \prod_{j\in[\ell]} v_j^{a_{ij}}$ [Symb] *for every* $i \in [t]$.

We actually care for the rank of the canonical-form matrix of the terms output by a circuit $C$, which shows if there exists an output term which can be expressed as a product of powers of the other output terms. This would imply that if we know the $e$-th roots of the latter then we can compute the $e$-th root of the former. Jumping forward, we will exploit this property of the canonical-form matrix to see if a query is degenerated.

We are finally ready to define $\mathsf{Forger}_G$.

**Algorithm 4** ($\mathsf{Forger}_G$)

*Input:* $q = (n, e, h, \{\sigma_i\}_{i\in[t]})$, *where* $n$, $e$ *and* $\{\sigma_i\}_{i\in[t]}$ *are integers, and* $h$ *is an oracle-aided circuit.*

*Operation:*

1. *If* $e \notin \text{P}$, $|h|\,(= |\text{desc}(h)|) > t$ *or for some* $i \in [t]$ $h^G(i) \notin \mathbb{Z}_n^*$ *or* $h^G(i) \not\equiv \sigma_i^e$ $[G_n]$, *return* $\perp$.
2. *Let* $M = M^{G,n,H}(\{h(i)\}_{i\in[t]})$ *according to Definition 7, where* $H$ *is the oracle-aided circuit that first evaluates* $h^G(1), \ldots, h^G(t)$ *and then queries* $G_n$ *on the answers (say asking for their inverses).*
   *If* $\text{rank}_e M < t$, *return* $\perp$.
3. *Return* $(h^G(0))^d$ $[G_n]$, *where* $d = e^{-1} \mod \phi(n)$.

That is, $\mathsf{Forger}_G$ first checks that $\{\sigma_i\}_{i\in[t]}$ are valid signatures for the messages $\{1, \ldots, t\}$ (with respect to $G$ and the public key $(n, e, h)$) and that forging a signature for this public key is not easy (reflected by $\text{rank}_e M = t$). If satisfied, $\mathsf{Forger}_G$ forges a signature for 0.

Below we describe the PPT $F$ that uses $\mathsf{Forger}_G$ for breaking the security of $\Sigma^G$.

### 3.2 The Breaker $F$

The strategy of the algorithm $F$ that uses $\mathsf{Forger}$ for breaking the security of $\Sigma^G$ is simple: on input $(n, e, h)$ it would like to use $\mathsf{Forger}$ on $(n, e, h, \{\sigma_i = \mathsf{Sign}^G(n, e, i)\}_{i\in[t]})$ to forge the signature of 0. It might be the case, however, that $\mathsf{Forger}$ returns bottom on such input. Hence, $F$ first checks by himself (without using $\mathsf{Sign}$ or $\mathsf{Forger}$) whether $\mathsf{Forger}$ will return bottom on this input. If positive, it uses a straightforward approach (see below) for forging a message $k \in [t]$, without using $\mathsf{Forger}$ at all.

**Algorithm 5 ($F$)**
*Input:* $pk = (n, e, h)$
*Oracles:* $G \in \mathcal{G}_n$, $\mathsf{Sign}^G(sk, pk, \cdot)$ *and* $\mathsf{Forger}_G$.
*Operation:*

1. *Let* $t = |h|$ *and let* $M = M^{G,n,H}(\{h^G(i)\}_{i\in[t]})$ *according to Definition 7, where* $H$ *is as in Algorithm 4 (with respect to this* $h$ *and* $t$*).*
2. *If* $\mathrm{rank}_e(M) = t$, *return* $\mathsf{Forger}_G(n, e, h, \{\mathsf{Sign}^G(sk, pk, i)\}_{i\in[t]})$.
   *Otherwise,*
   (a) *Using Gaussian Elimination find* $k \in [t]$ *and a set* $\{\lambda_i \in [e]\}_{i\in[t]\setminus\{k\}}$, *such that for every* $j \in [\ell]$ *it holds that* $M_{kj} \equiv \sum_{i\in[t]\setminus\{k\}} \lambda_i \cdot M_{ij} \mod e$.
   (b) *Let* $\gamma = \prod_{j\in[\ell]} v_j^{(M_{kj} - \sum_{i\in[t]\setminus\{k\}} \lambda_i \cdot M_{ij})/e} \ [G_n]$, *where* $\{v_i\}_{i\in[\ell]} = \mathrm{HardWired}_{H,G,n}$ *(see Definition 5).*
   (c) *For every* $i \in [t] \setminus \{k\}$, *let* $\sigma_i = \mathsf{Sign}^G(sk, pk, i) \ \ (\equiv h^G(i)^d \ \ [G_n])$.
   (d) *Return* $\sigma_k = \gamma \cdot \prod_{i\in[t]\setminus\{k\}} \sigma_i^{\lambda_i} \ \ [G_n]$.

The following lemma is immediate, but its proof is omitted and can be found in the full version of this paper [14].

**Lemma 3.** *For every* $G \in \mathcal{G}$, $F^{G,\mathsf{Forger}_G}$ *breaks the security of* $\Sigma^G$.

### 3.3 The Emulator

Our task is to emulate a family of circuits $\{C_k\}$ with oracle access to $G \in \mathcal{G}$ and $\mathsf{Forger}_G$, using only oracle access to $G$. We assume without loss of generality that $|C_k| \geq k$ (otherwise we emulate a padded version of this family) and omit $k$ from the input parameter list of the emulator. We also assume without loss of generality that before calling $\mathsf{Forger}_G$ on input $(n, e, h, \{\sigma_i\}_{i\in[t]})$, $C_k$ first query $G$ on $\{\sigma_i\}$ (otherwise, we will emulate the circuit $C'_k$ that does so).

Given a circuit $C$, $\mathsf{Emul}^G(\mathrm{desc}(C))$ emulates the execution of a circuit $C^{G,\mathsf{Forger}_G}$ by forwarding the $G$-calls to $G$, and answering the $\mathsf{Forger}_G$-calls using the following method: let $q = (n, e, h, \{\sigma_i\}_{i\in[t]})$ be a query that $C$ makes to $\mathsf{Forger}_G$, $\mathsf{Emul}$ first checks whether $\mathsf{Forger}_G$ returns bottom on this call (which it can do efficiently), and if positive returns bottom to $C$ as well. Otherwise, $\mathsf{Emul}$ uses the query $q$ and the description of $C$ to factor $n$, and then uses this factorization to answer the query efficiently.

The interesting question is how can $\mathsf{Emul}$ use such a pair $(C, q)$ to factor $n$ efficiently? Let $H$ and $M^H = M^{G,n,H}(\{h(i)\}_{i\in[t]})$ as computed by $\mathsf{Forger}_G(q)$, and let $M^{(H;C)} = M^{G,n,(H;C)}(\{\sigma_i\}_{i\in[t]}) \in \mathbb{Z}_{t\times\ell'}$, where the circuit $(H;C)$ first evaluates $H$ and then $C$.[9] Namely, $M^H$ represents the canonical form of $\{h(i)\}_{i\in[t]}$ induced by the (stand alone) computation of $H$, where $M^{(H;C)}$ represents the canonical form of the "signatures" $\{\sigma_i\}_{i\in[t]}$ induced by the computation of $(H;C)$. Since $(H;C)$ first starts by computing $H$, it follows that every hardwired term $u \in \text{HardWired}_{H,G,n} \cap \text{HardWired}_{(H;C),G,n}$ has the same index with respect to both ordered sets $\text{HardWired}_{H,G,n}$ and $\text{HardWired}_{(H;C),G,n}$. Hence, the promise that $\sigma_i^e \equiv h(i)\ [G_n]$ for every $i \in [t]$, yields the following with respect to $\{v_i\}_{i\in[\ell']} = \text{HardWired}_{(H;C),G,n}$ :

$$\prod_{j\in[\ell]} v_j^{M^H_{ij}} \equiv \prod_{j\in[\ell']} (v_j^{M^{(H;C)}_{ij}})^e \ \ [G_n],$$

for every $i \in [t]$. Since $G_n$ is selected at random, (at least intuitively) $C$ could have satisfied the above equations only if they hold regardless of the choice of $G_n$. Namely, it is the case that

$$\sum_{j\in[\ell]} M^H_{ij} \equiv e \cdot \sum_{j\in[\ell']} M^{(H;C)}_{ij} \mod \phi(n) \tag{1}$$

for every $i \in [t]$. On the other hand, the assumption that $\mathsf{Forger}_G(q) \neq \perp$ yields that $\text{rank}_e M^H = t$. Therefore, Equation (1) is "far" from being satisfied modulo $e$. In our proof we show how to use this inconsistency to find a multiple of $\phi(n)$, and thus to factor $n$.

The following description of $\mathsf{Emul}$ realizes the above discussion. We start by recalling the following known factoring algorithms. The first one is useful for small $n$'s (for which the above discussion does not hold), and the second one factors arbitrary larger $n$, given a multiple of $\phi(n)$ as an advice.

**Theorem 6 (factoring small numbers, [11,34]).** *There exists a procedure* $\mathsf{Sef}$ *that on input* $n \in \mathbb{N}$*, runs in time* $2^{O(\sqrt{\log n \log\log n})}$ *and factors* $n$ *with constant probability.*

**Lemma 4 (factoring using multiple of $\phi(n)$).** *We say that* $z = (z_1, z_2) \in \mathbb{Z}\times\mathbb{N}$ *is a* factoring advice *for* $n \in \mathbb{N}$*, if* $z_1^{\lceil \log n\rceil} \cdot \prod_{p\in \mathrm{P}:\ p<z_2} p^{\lceil \log n\rceil}$ *is a non-zero multiple of* $\phi(n)$*.*

*There exists a procedure* $\mathsf{Factor}$ *that on input* $(n, z_1, z_2)$*, runs in time* $\text{poly}(z_2)\cdot\text{poly}(\log|nz_1|)$*, and factors* $n$ *with constant probability, assuming that* $z = (z_1, z_2)$ *is a factoring advice for* $n$*.*

*Proof.* We use the following known algorithm due to Miller, [24].

**Theorem 7 (Miller's algorithm [24,34]).** *There exists a procedure that on input* $n \in \mathbb{N}$ *and* $\mu \in \mathbb{Z}$*, runs in time* $\text{poly}(\log|n\mu|)$*, and if* $\mu$ *is a non-zero multiple of* $\phi(n)$*, it factors* $n$ *with constant probability.*

[9] Recall that we allow circuits to have a predetermined evaluating order.

By definition $\mu = z_1^{\lceil \log n \rceil} \cdot \prod_{p \in \mathrm{P}:\, p < z_2} p^{\lceil \log n \rceil}$ is a non-zero multiple of $\phi(n)$. Thus, Miller's algorithm on input $(n, \mu)$, runs in time $\mathrm{poly}(\log |n\mu|) = \mathrm{poly}(z_2 \cdot \log |nz_1|)$ and factors $n$ with constant probability. Finally, note that $\mu$ is easily computable in time $\mathrm{poly}(z_2, \log n)$.

We are now finally ready to define $\mathsf{Emul}$.

**Algorithm 8** ($\mathsf{Emul}$)
*Input: The description of an oracle-aided circuit $C$.*
*Oracle: $G \in \mathcal{G}$.*
*Operation:*
*Emulate $C^G$ while on every query $q = (n, e, h, \{\sigma_i\}_{i \in [t]})$ to $\mathsf{Forger}_G$, return the following value to $C$:*

1. *If $\mathsf{Forger}_G$ would return $\perp$ on $q$, return $\perp$ as well (and continue to the next query). Else,*
2. *Try to factor $n$ by doing the following for $|C|$ times:*
   *If $n \leq |C|^{\frac{\log |C|}{\log \log |C|}}$, execute $\mathsf{Sef}(n)$.*
   *Otherwise, execute $\mathsf{Factor}(n, \det(Q_{C,G,q}), |C|^4)$, where $Q_{C,G,q}$ is according to Definition 8.*
3. *If factoring of $n$ is successful, return $h^G(0)^d \ [G_n]$, where $d = e^{-1} \bmod \phi(n)$.*
   *Otherwise, abort.*

The matrix $Q_{C,G,q}$ is defined as follows:

**Definition 8 (query matrix).** *Let $C$ be an oracle-aided circuit, $G \in \mathcal{G}$ and let $q = (n, e, h, \{\sigma_i\}_{i \in [t]})$ be the query asked by $C^{G, \mathsf{Forger}_G}$ to $\mathsf{Forger}_G$. The matrix $Q_{C,G,q} \in \mathbb{Z}_{t \times t}$ is defined as follows:*

1. *If $\mathsf{Forger}_G(q) = \perp$, set $Q_{C,G,q} = 0_{t \times t}$.*
   *Otherwise:*
2. *Let $M^H = M^{G,n,H}(\{h(i)\}_{i \in [t]})$ according to Definition 7, where $H$ is as in Algorithm 4 with respect to this $h$ and $t$. (Since $\mathsf{Forger}_G(q) \neq \perp$, the matrix $M^H$ is well defined and of rank $t$.)*
3. *Let $\mathcal{I} \subseteq [\ell]$ be the first subset of size $t$ (from hereafter we assume some arbitrary order on such sets) with $\mathrm{rank}_e(M^H_{\mathcal{I}}) = t$.*[10]
4. *Let $M^{(H;C)} \in \mathbb{Z}_{t \times \ell'}$ be the matrix $M^{G,n,(H;C)}(\{\sigma_i\}_{i \in [t]})$ according to Definition 7, where $(H;C)$ is the circuit that first evaluates $H$ and then evaluates $C$.*
5. *Set $Q_{C,G,q} = M^H_{\mathcal{I}} - e \cdot M^{(H;C)}_{\mathcal{I}}$.*

Note that in the code of $\mathsf{Emul}$ if $\mathsf{Sef}$ is called, and thus $n$ is small, then it runs in time $\mathrm{poly}(|C|)$. In addition, the running time of $\mathsf{Factor}$, if called, is also in $\mathrm{poly}(|C|)$. Thus, $\mathsf{Emul}$ runs in polynomial time.

[10] Remember that $M^H_{\mathcal{I}} \in \mathbb{Z}_{t \times t}$ is the restriction of $M^H$ to the columns in $\mathcal{I}$.

Moreover, it is clear that the only case where the output of $\mathsf{Emul}^G(\mathrm{desc}(C))$ differs from the output of $C^G$ is when the former aborts. This means that for some query of $C$ to $\mathsf{Forger}$, the latter would not return $\perp$, but either (1) $\mathsf{Sef}$ failed, or (2) $z$ was a factoring advice but $\mathsf{Factor}$ failed, or (3) $z$ was not a factoring advice for $n$. As the first two cases happen with negligible probability (by Theorem 6 and Lemma 4), we only have to prove that the latter happens with negligible probability.

This is formally done in the following lemma, whose proof (done via the "short description paradigm") can be found in the full version of this paper [14].

**Lemma 5.** *A query $q = (n, \cdot)$ to* $\mathsf{Forger}$ *made by $C^{G\in\mathcal{G},\mathsf{Forger}_G}$ is* unexpected*, if*

- $\mathsf{Forger}_G(q) \neq \perp$,
- $n > |C|^{\frac{\log|C|}{\log\log|C|}}$, *and*
- $(\det(Q_{C,G,q}), |C|^4)$ *is* not *a factoring advice for $n$, where $Q_{C,G,q}$ is according to Definition 8.*

*The following holds for any oracle-aided circuit $C$:*

$$\mathsf{Pr}_{G\leftarrow\mathcal{G}}[C^{G,\mathsf{Forger}_G} \textit{ asks } \mathsf{Forger} \textit{ an unexpected query}] \leq \delta(|C|),$$

*where $\delta(|C|) = 2^{-\log^2|C|}$.*

### 3.4 Putting It Together

*Proof (of Lemma 2).* Lemma 3 yields that $F^{G,\mathsf{Forger}_G}$ breaks the security of $\Sigma^G$ with respect to every $G \in \mathcal{G}$, so it is left to prove that $\mathsf{Emul}^G(C_k)$ emulates $C_k^{G,\mathsf{Forger}_G}$ well.

Recall that $|C_k| \in \mathrm{poly}(k)$, and that we assume without loss of generality that $|C_k| \geq k$. Theorem 6 and Lemma 4 yield that $\mathsf{Emul}(C_k)$ answers all "expected" queries of $C_k$ to $\mathsf{Forger}$ with probability $1 - |C_k| \cdot 2^{-\Omega(k)} = 1 - \mathrm{neg}(k)$, where Lemma 5 yields that $C_k$ asks unexpected queries with only negligible probability over the choice of $G \in \mathcal{G}$. Hence, with save but negligible probability, $\mathsf{Emul}^G(C_k)$ emulates $C_k^{G,\mathsf{Forger}_G}$ correctly.

**Acknowledgments.** We thank Nir Bitansky, Thomas Holenstein and Ilya Mironov for very helpful conversations.

## References

1. Aggarwal, D., Maurer, U.: Breaking RSA Generically Is Equivalent to Factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 36–53. Springer, Heidelberg (2009)
2. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)

3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Boldyreva, A., Fischlin, M.: Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005)
6. Brown, J., González Nieto, J.M., Boyd, C.: Efficient CCA-Secure Public-Key Encryption Schemes from RSA-Related Assumptions. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 176–190. Springer, Heidelberg (2006)
7. Canetti, R., Goldreich, O., Halevi, S.: On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 40–57. Springer, Heidelberg (2004)
8. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. JACM: Journal of the ACM, 51 (2004)
9. Coron, J.-S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
10. Cramer, R., Shoup, V.: Signature schemes based on the strong rsa assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
11. Dixon, J.D.: Asymptotically fast factorization of integers. Mathematics of Computation 36, 255–260 (1981)
12. Dodis, Y., Oliveira, R., Pietrzak, K.: On the Generic Insecurity of the Full Domain Hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
13. Dodis, Y., Reyzin, L.: On the Power of Claw-Free Permutations. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 55–73. Springer, Heidelberg (2003)
14. Dodis, Y., Haitner, I., Tentes, A.: On the instantiability of hash-and-sign rsa signatures. ePrint, `http://eprint.iacr.org/2011/087`
15. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM Journal on Computing 35(1), 217–246 (2005)
16. Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures without the Random Oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
17. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 305–313. IEEE Computer Society (2000)
18. Goldwasser, S., Tauman-Kalai, Y.: On the (in)security of the fiat-shamir paradigm. In: Proceedings of the 44th Annual Symposium on Foundations of Computer Science (FOCS), pp. 102–113. IEEE Computer Society (2003)
19. Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols – A tight lower bound on the round complexity of statistically-hiding commitments. In: Proceedings of the 48th Annual Symposium on Foundations of Computer Science (FOCS), pp. 669–679. IEEE Computer Society (2007)
20. Haitner, I., Holenstein, T.: On the (Im)Possibility of Key Dependent Encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (2009)
21. Hofheinz, D., Jager, T., Kiltz, E.: Short Signatures From Weaker Assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011)

22. Hohenberger, S., Waters, B.: Short and Stateless Signatures from the RSA Assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
23. Maurer, U.M.: Abstract models of computation in cryptography. In: IMA Int. Conf., pp. 1–12 (2005)
24. Miller, G.L.: Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences 13(3), 300–317 (1976)
25. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. MATHNASUSSR: Mathematical Notes of the Academy of Sciences of the USSR, 55 (1994)
26. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
27. Paillier, P.: Impossibility Proofs for RSA Signatures in the Standard Model. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 31–48. Springer, Heidelberg (2006)
28. Paillier, P., Vergnaud, D.: Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005)
29. Paillier, P., Villar, J.L.: Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 252–266. Springer, Heidelberg (2006)
30. Pietrzak, K.: Compression from Collisions, or Why CRHF Combiners Have a Long Output. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 413–432. Springer, Heidelberg (2008)
31. Rivest, R.L., Shamir, A., Adelman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
32. RSA Laboratories, Redwood City, California. PKCS #1: RSA Encryption Standard (November 1993)
33. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
34. Shoup, V.: Computational Introduction to Number Theory and Algebra. Cambridge University Press (2005)
35. Wee, H.: One-Way Permutations, Interactive Hashing and Statistically Hiding Commitments. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 419–433. Springer, Heidelberg (2007)