On the Instantiability of Hash-and-Sign RSA Signatures

Yevgeniy Dodis* Iftach Haitner† Aris Tentes‡

December 29, 2011

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.

*Department of Computer Science, New York, University. E-mail: `dodis@cs.nyu.edu`.

†School of Computer Science, Tel Aviv University. E-mail: `iftachh@cs.tau.ac.il`.

‡Department of Computer Science, New York, University. E-mail: `tentes@cs.nyu.edu`.

Contents

1 Introduction

Bellare and Rogaway [2] 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) [2] and RSA *Optimal Asymmetric Encryption Padding* scheme (RSA-OAEP) [3]. 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 [32] 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 [33]. In its "textbook" form, the signature σ of the message m is simply $\sigma = m^d \bmod n$, which can be verified by checking if $\sigma^e \equiv m \mod n$, where e is the public RSA exponent, and $d = e^{-1} \bmod \phi(n)$. Of course, the textbook variant is completely insecure, as any σ 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 [2] 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 [7, 8, 27, 17, 4]. 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 [28] 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 [15, 10, 21], 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 [2, 38]).

More recently, Paillier [28] 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, 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 [28], 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 Paillier [28] 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, 28]. To motivate this result, we start by describing our already mentioned positive result.

Theorem 1.1 (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*

[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 [2].

[2] With a different motivation, the same result was independently obtained by [20].

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 Section 6, 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 [28]). 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 1.2 (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 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 3) using the notion of *generic groups* [35, 26, 24]. 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 1.2 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 1.1, the standard model constructions of [15, 10, 21], and the random-oracle based analysis of [2, 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, 28]. 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 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 R must already "know" (so that Forger never helps R compute something which R does not know to begin with, potentially helping 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 Dodis et al. [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 π 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 σ_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 π 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 and Trevisan [14] short description paradigm, described below, to the generic groups realm.[4]

1.2.1 The Gennaro and Trevisan short description paradigm and its adaption to generic groups

Loosely, Gennaro and Trevisan [14] show 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 and Trevisan paradigm

[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 ℓ.

[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] ([1], however, also prove it over "generic rings").

(from now on, the GT paradigm) has several extensions (e.g., [16, 37, 19, 18, 31]), all are given in the random functions realm.

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 π (which defines the group $\pi(\mathbb{Z}_n^*)$). Since compressing π 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 π). 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 π. 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 π. 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, 36].

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 [29] 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 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 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 R to use some "hidden values" related to the assumption that R is trying to break, "algebraic" reduction do not allow this flexibility. Thus, much of the technical difficulties in

[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.

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 [29] 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 [30] and Brown et al. [6], showed uninstantiability results analogous to already-mentioned RSA signature result of Paillier [28].

Paper Organization

Section 2 contains basic notation and some basic linear algebra facts, where in Section 3 we formally define RSA-FDH and its security in the generic group model. Our main result, regarding the impossibility of existentially unforgeable RSA-FDH against unbounded number of signing queries, is proven in Section 4, where in Section 5 we prove our main technical lemma using the GT short description paradigm. Finally, in Section 6 we present our construction of an existentially unforgeable RSA-FDH scheme against a bounded number of signing queries.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables and matrices, and lowercase for values. Given a random variable X, $X^{(t)} = (X_1, \ldots, X_t)$ consists of t independent copies of X, where for a set $\mathcal{S}$, $\mathcal{S}^{(t)} = (\mathcal{S}_1, \ldots, \mathcal{S}_t)$ denotes the t'th direct product of $\mathcal{S}$. For integer $n \in \mathbb{N}$, we let $[n] = \{1, \cdots, n\}$. Given a matrix $M \in \mathcal{U}_{t\times q}$ and a set of indices $\mathcal{I} \subseteq [q]$, the matrix $M_{\mathcal{I}} \in \mathcal{U}_{t\times|\mathcal{I}|}$ denotes the restriction of M to the columns in $\mathcal{I}$.

We let poly denote the set of polynomial, and let PPT denote the set of probabilistic algorithms (i.e., Turing machines) that run in *strict* polynomial time. A function $\mu\colon \mathbb{N} \to [0, 1]$ is *negligible* if $\mu(n) = n^{-\omega(1)}$, where neg denotes the family of negligible functions. Throughout the text we sometimes abuse notation and view poly and neg also as arbitrary members of the families they represent (e.g., we write $f(n) = \mathrm{neg}(n)$ to denote $f \in \mathrm{neg}$ and $f(n) > \mathrm{neg}(n)$ for $f \notin \mathrm{neg}$).

Given a random variable X taking values in a finite set $\mathcal{U}$, we write $x \leftarrow X$ to indicate that x is selected according to X. Similarly given a set $\mathcal{S} \subseteq \mathcal{U}$, we let $s \leftarrow \mathcal{S}$ denote that s is selected according to the uniform distribution on $\mathcal{S}$. We adopt the convention that when the same random variable occurs several times in an expression, all occurrences refer to a single sample. For example, $\mathsf{Pr}[f(X) = X]$ is defined to be the probability that when $x \leftarrow X$, we have $f(x) = x$. We write U_n to denote the random variable distributed uniformly over $\{0, 1\}^n$. A distribution ensemble $\mathcal{D} = \{D_k\}_{k\in\mathbb{N}}$ is of polynomial-length, if every element of D_k is described using $\mathrm{poly}(k)$ bits. The statistical distance of two distributions P and Q over $\mathcal{U}$ is defined as

$$\mathrm{SD}(P, Q) := \frac{1}{2}\sum_{u\in\mathcal{U}} |P(u) - Q(u)|$$

We let P denote the prime numbers, and for $n \in \mathbb{N}$ let $\mathbb{Z}_n^*$ denote the group of elements in $[n]$ that are relatively prime to n, where multiplication mod n is the group operation. Let C be a circuit, then by $|C|$ we denote the length of the binary description of C.

2.2 Useful Linear Algebra Facts

Definition 2.1. *Let M be an integer matrix with rows $\{v_1, \ldots, v_t\}$, and let $e \in \mathrm{P}$.*

- *The rows of M are* linearly dependent *if there exist not all zeros real numbers $\{a_i\}_{i \in [t]}$, such that $\sum_{i \in [t]} a_i \cdot v_i = 0$.*
- *The rows of M are* linearly dependent modulo e, *if there exist not all zeros numbers in $\mathbb{Z}_e$ $\{a_i\}_{i \in [t]}$, such that $\sum_{i \in [t]} a_i v_i = 0 \mod e$.*
- $\mathrm{rank}(M)$ *is the maximum number of rows of M that are linearly independent.*
- $\mathrm{rank}_e(M)$ *is the maximum number of rows of M that are linearly independent modulo e.*

The rank of any integer matrix M can be efficiently computed using Gaussian Elimination, which computes the reduced row Echelon form $\widetilde{M}$ of M such that $\mathrm{rank}(M)$ is the number of non zero rows of $\widetilde{M}$. Similarly we can compute $\mathrm{rank}_e(M)$, but now every computation is done $\bmod e$. Notice that the latter is a well defined computation since every element in $\mathbb{Z}_e$ has an inverse. Moreover, the analogue of many properties of $\mathrm{rank}(M)$ are also true for $\mathrm{rank}_e(M)$, because working $\bmod e$ simply means working in another field ($\mathbb{Z}_e$ instead of $\mathbb{Q}$ or $\mathbb{R}$). In particular, we have the following:

Fact 2.2. *The following holds for every $e \in \mathrm{P}$:*

- *Let $M \in \mathbb{Z}_{t \times \ell}$. If $\mathrm{rank}_e(M) = s$, then there exists a (polynomial-time computable) submatrix $M' \in \mathbb{Z}_{s \times s}$ of M with $\det(M') \not\equiv 0 \mod e$.*
- *Let $M \in \mathbb{Z}_{s \times s}$, then $\mathrm{rank}_e(M) = s$ iff $\det(M) \not\equiv 0 \mod e$.*

3 RSA-FDH in the Generic Group Model

We start by recalling the standard notion of RSA-FDH signature scheme.

3.1 RSA-FDH Signature Scheme

Definition 3.1 (RSA-FDH)**.** *An* RSA-FDH *signature scheme Σ consists of the following triplet* (KeyGen, Sign, Verify) *of polynomial-time algorithms:*

- *On security parameter 1^k,* KeyGen *outputs a "public key" (n, e, h), where n is a product of two primes, e is a element in $\mathbb{Z}^*_{\phi(n)}$ and h is a (hash) function, represented as an oracle-aided circuit, mapping values to $\mathbb{Z}^*_n$, and a "secret key" $d = e^{-1} \bmod \phi(n)$.*
- *On 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,* Sign *outputs the "signature" $h(m)^d \bmod n$.*
- *On 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 $\sigma \in \mathbb{Z}^*_n$,* Verify *outputs one iff $\sigma^e \equiv h(m) \mod n$.*

Now let us see what it means that an RSA-FDH signature scheme is existentially unforgeable under unbounded and bounded chosen message attack (EU-CMA-secure and t-EU-CMA-secure):

Definition 3.2 (security of RSA-FDH)**.** *An oracle-aided algorithm* F breaks the security *of an* RSA-FDH *signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, if*

$$\begin{aligned}\mathsf{Pr}_{(sk,pk)\leftarrow\mathsf{KeyGen}(1^k)}[(m,\sigma) \leftarrow F^{\mathsf{Sign}(sk,pk,\cdot)}(pk)\colon& \\ \mathsf{Verify}(\sigma,m,pk) = 1 \wedge \mathsf{Sign} \textit{ was not queried on } (sk,pk,m)] > \operatorname{neg}(k)&\end{aligned} \tag{1}$$

A signature scheme Σ *is* EU-CMA*-secure, if no (oracle-aided)* PPT *breaks its security, where* Σ *is* t-EU-CMA*-secure, if no* PPT *breaks its security when restricted to query* Sign *at most* $t(k)$ *times.*

Remark 3.3 (Discussion)**.** *Definition 3.1 allows the hash function* h *to be chosen as part of the public key, where* h *needs to be described as a circuit.*[6] *In practice, however, a fixed hash function (e.g., SHA-1) defined over any string is used. Since any secure scheme (according to Definition 3.2) of the type used in practice trivially yields a secure scheme of the type considered in Definition 3.1, for the sake of impossibility results it suffices to consider Definition 3.1. Furthermore, a positive result according to Definitions 3.1 and 3.2, can be easily extended to output hash functions defined over all strings: first hash the message using a secure collision-resistant hash function (assuming such function exists), and then apply the bounded length scheme.*

Additional restriction of Definition 3.1 is that it requires the range of the hash function h *to be a subset of* $\mathbb{Z}_n^*$*, where in practice the range of* h *is an arbitrary subset of* $\mathbb{Z}_n$*. Notice, however, that it is easy to forge the signature of a given message* m *with* $h(m) \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$*: if* $h(m) = 0$*, its signature is simply* 0*, otherwise* $\gcd(h(m), n)$ *implies a factorization of* n*, which in turn can be used to forge the signature of any message. It follows that by modifying the hash function used in a given* RSA-FDH *scheme to set* $h(x) = 1$ *whenever* $h(x) \notin \mathbb{Z}_n^*$*, one does not hurt the security of the scheme. In particular, for the sake of impossibility results it suffices to consider hash functions whose range is a subset of* $\mathbb{Z}_n^*$*.*

In the following we first formally define what we mean by generic group model, and then extend Definitions 3.1 and 3.2 to this model.

3.2 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 1.2). In the *generic algorithm model* due to Maurer [24], "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. Namely, our model can be viewed as a model for proving impossibility results in the generic algorithm model.

[6] Alternatively, h can described as a Turing Machine running in time $\operatorname{poly}(k)$, where k being the security parameter.

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 π. 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).

3.3 RSA-FDH Signature Schemes in the Generic Group Model

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

Definition 3.4 (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, 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, 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 $\sigma \in \mathbb{Z}_n^*$, Verify^G outputs one iff $\sigma^e \equiv h^G(m) \quad [G_n]$.*

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

3.3.1 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 3.5 (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*

$$\Pr_{G\leftarrow\mathcal{G},(sk,pk)\leftarrow\mathsf{KeyGen}^G(1^k)}[(m,\sigma) \leftarrow F^{G,\mathsf{Sign}^G(sk,pk,\cdot)}(pk) \colon \mathsf{Verify}^G(\sigma, m, pk) = 1 \wedge \mathsf{Sign} \textit{ was not queried on } (sk, pk, m)] > \operatorname{neg}(k)$$

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* Sign *at most $t(k)$ times.*

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 3.6 (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* X*, if there exists an oracle-aided* PPT *R such that if* X *is true, then the following holds: let F be a (possibly unbounded) adversary that breaks the security of $\Sigma^{\mathcal{G}}$ (see Definition 3.5), then for any* PPT Emul *there exists a polynomial-length distribution ensemble $\mathcal{D} = \{D_k\}_{k\in\mathbb{N}}$ such that*

$$\mathrm{SD}\left((x, R^{G,F^G}(1^k,x)), (x, \mathsf{Emul}^G(1^k,x))\right)_{G\leftarrow\mathcal{G}, x\leftarrow D_k} > \mathrm{neg}(k).$$[7]

Remark 3.7 (A black-box proof implies a weakly black-box proof)**.** *Assuming that* X *is true, the above intuitively asks that a security breach of $\Sigma^{\mathcal{G}}$ implies that a (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 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* Emul *such that $(x, \mathsf{Emul}^G(1^k,x))_{G\leftarrow\mathcal{G}, x\leftarrow D_k}$ is (even computationally) close to ξ_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 Emul *such that $(x, \mathsf{Emul}^G(1^k,x))_{G\leftarrow\mathcal{G}, x\leftarrow D_k}$ is (even computational) close to ξ_k. Namely, there is no weakly black-box proof of security for $\Sigma^{\mathcal{G}}$* based *on factoring.*[8]

For completeness, we give the following natural adaptation of the RSA assumption to the generic group model.

Definition 3.8 (The RSA assumption in the generic group model)**.** *There exists an oracle aided* PPT Gen*, which on input 1^k outputs (n, e), where $n \in \mathbb{N}$ is a product of two primes and* $\gcd(e, \phi(n)) = 1$ *such that the following holds for any oracle-aided* PPT *A:*

$$\mathsf{Pr}_{G\leftarrow\mathcal{G},(n,e)\leftarrow\mathsf{Gen}^G(1^k), x\leftarrow\mathbb{Z}_n^*}\left[(A^G(1^k,n,e,x))^e \equiv x \quad [G_n]\right] = \mathrm{neg}(k).$$

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

[8]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).

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

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

Theorem 4.1 (Theorem 1.2, 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)* X*, then* X *is false.*

The proof of Theorem 4.1 immediately follows from the next lemma:

Lemma 4.2. *Let* $\Sigma^{\mathcal{G}}$ *be as in Theorem 4.1, then there exist a family of oracles* $\mathsf{Forger} = \{\mathsf{Forger}_G\}_{G \in \mathcal{G}}$ *and oracle-aided* PPT*'s* F *and* Emul*, such that the following hold:*

1. *For every* $G \in \mathcal{G}$*,* F^{G,Forger_G} *breaks the security of* Σ^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 4.2, let us first use it for proving Theorem 4.1.

Proof of Theorem 4.1. 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) X and let R be the algorithm guaranteed by this proof. Let Emul be the algorithm guaranteed by Lemma 4.2 with respect to $\Sigma^{\mathcal{G}}$. Lemma 4.2 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 X is false. $\square$

The rest of this section is devoted for proving Lemma 4.2. We find it more convenient, however, to prove a variant of Lemma 4.2 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 4.3 (non uniform variant of Lemma 4.2)**.**

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* 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 4.2. In Section 4.1 we define the family of oracles Forger and the efficient algorithm F that uses Forger to break any RSA-FDH scheme, in Section 4.2 we define the emulator Emul, where in Section 4.3 we put things together to prove Lemma 4.3.

4.1 The Forger

Recall (see Section 1.2) that 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 $\{h(i)\}_{i\in[t]}$,[9] as a function of the group queries done through 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 4.4 (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 ε (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 4.5 (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* $\mathrm{Terms}_{C,G,n}$*, 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* $\mathrm{HardWired}_{C,G,n}$ *are those element inside* $\mathrm{Terms}_{C,G,n}$ *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* $\mathrm{HardWired}_{C,G,n}$*). We assume that the elements of each of the above sets are ordered according to the evaluation order.*

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

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

Definition 4.6 (canonical form)**.** *Let C, G and n be as in Definition 4.5. The* canonical form *of $u \in \mathrm{Terms}_{C,G,n}$ with respect to (C,G,n), denoted $\mathrm{Can}_{C,G,n}(u)$, is recursively defined as follows:*

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

Let $\{v_i\}_{i\in[\ell]} = \mathrm{HardWired}_{C,G,n}$. Note that the canonical form of any $u \in \mathrm{Terms}_{C,G,n}$ 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 4.7 (canonical-form matrix)**.** *Let C, G and n be as in Definition 4.5, let $\{v_i\}_{i\in[\ell]} = \mathrm{HardWired}_{C,G,n}$ and let $\mathcal{W} = \{u_i\}_{i\in[t]} \subseteq \mathrm{Terms}_{C,G,n}$. 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 $\mathrm{Can}_{C,G,n}(u_i) = \prod_{j\in[\ell]} v_j^{a_{ij}}$ [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 Forger_G.

Algorithm 4.8 (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 $e \notin \mathrm{P}$, $|h| (= |\mathrm{desc}(h)|) > t$ 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 4.7, 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 $\mathrm{rank}_e M < t$, return $\perp$.
3. *Return $(h^G(0))^d$ $[G_n]$, where $d = e^{-1} \mod \phi(n)$.*

That is, 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 $\mathrm{rank}_e M = t$). If satisfied, Forger_G forges a signature for 0.

Below we describe the PPT F that uses Forger_G for breaking the security of Σ^G.

4.1.1 The breaker F

The strategy of the algorithm F that uses Forger for breaking the security of Σ^G is simple: on input (n, e, h) it would like to use 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 Forger returns bottom on such input. Hence, F first checks by himself (without using Sign or Forger) whether 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 Forger at all.

Algorithm 4.9 (F)**.**

Input: $pk = (n, e, h)$

Oracles: $G \in \mathcal{G}_n$, $\mathsf{Sign}^G(sk, pk, \cdot)$ *and* Forger_G.

Operation:

1. *Let* $t = |h|$ *and let* $M = M^{G,n,H}(\{h^G(i)\}_{i \in [t]})$ *according to Definition 4.7, where* H *is as in 4.8 (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* $\gamma = \prod_{j \in [\ell]} v_j^{(M_{kj} - \sum_{i \in [t] \setminus \{k\}} \lambda_i \cdot M_{ij})/e} \quad [G_n]$, *where* $\{v_i\}_{i \in [\ell]} = \mathrm{HardWired}_{H,G,n}$ *(see Definition 4.5).*

   (c) *For every* $i \in [t] \setminus \{k\}$, *let* $\sigma_i = \mathsf{Sign}^G(sk, pk, i) \quad (\equiv h^G(i)^d \quad [G_n])$.

   (d) *Return* $\sigma_k = \gamma \cdot \prod_{i \in [t] \setminus \{k\}} \sigma_i^{\lambda_i} \quad [G_n]$.

The following claim is immediate.

Claim 4.10. *For every* $G \in \mathcal{G}$, F^{G, Forger_G} *breaks the security of* Σ^G.

Proof. Let (n, e, h) be the public key of $\Sigma^{\mathcal{G}}$ with respect to to some $G \in \mathcal{G}$ with $e \in \mathrm{P}$. Assume that $\mathrm{rank}(M) < t$ in the execution of F (otherwise the proof is immediate). In such a case, Step 3.(a) is guaranteed to succeed (and can be performed in polynomial time). It follows that

$$\begin{aligned}
\sigma_k^e &\equiv \gamma^e \cdot \prod_{i \in [t] \setminus \{k\}} h(i)^{\lambda_i} \quad [G_n] \\
&\equiv \gamma^e \cdot \prod_{i \in [t] \setminus \{k\}} \prod_{j \in [\ell]} v_j^{\lambda_i \cdot M_{ij}} \quad [G_n] \\
&\equiv \gamma^e \cdot \prod_{j \in [\ell]} v_j^{\sum_{i \in [t] \setminus \{k\}} \lambda_i \cdot M_{ij}} \quad [G_n] \\
&\equiv \gamma^e \cdot h(k) \cdot \prod_{j \in [\ell]} v_j^{-M_{kj} + \sum_{i \in [t] \setminus \{k\}} \lambda_i \cdot M_{ij}} \quad [G_n] \\
&\equiv \gamma^e \cdot h(k) \cdot \gamma^{-e} \quad [G_n] \\
&\equiv h(k) \quad [G_n].
\end{aligned}$$

Namely, σ_k^e is a valid signature of k, and since Sign^G was not asked on k, F breaks the security of $\Sigma^{\mathcal{G}}$ whenever $e \in \mathrm{P}$. $\square$

4.2 The Emulator

Our task is to emulate a family of circuits $\{C_k\}$ with oracle access to $G \in \mathcal{G}$ and 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 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,Forger_G} by forwarding the G-calls to G, and answering the Forger_G-calls using the following method: let $q = (n, e, h, \{\sigma_i\}_{i\in[t]})$ be a query that C makes to Forger_G, Emul first checks whether Forger_G returns bottom on this call (which it can do efficiently), and if positive returns bottom to C as well. Otherwise, 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 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.[10] 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 $u \in \mathrm{HardWired}_{H,G,n} \cap \mathrm{HardWired}_{(H;C),G,n}$ has the same index with respect to both ordered sets $\mathrm{HardWired}_{H,G,n}$ and $\mathrm{HardWired}_{(H;C),G,n}$. Hence, the promise that $\sigma_i^e \equiv h(i) \;\; [G_n]$ for every $i \in [t]$, yields the following with respect to $\{v_i\}_{i\in[\ell']} = \mathrm{HardWired}_{(H;C),G,n}$:

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

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

$$\sum_{j\in[\ell]} M_{ij}^H \equiv e \cdot \sum_{j\in[\ell']} M_{ij}^{(H;C)} \mod \phi(n) \tag{2}$$

for every $i \in [t]$. On the other hand, the assumption that $\mathsf{Forger}_G(q) \neq \perp$ yields that $\mathrm{rank}_e M^H = t$. Therefore, Equation (2) 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 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 4.11 (factoring small numbers, [11, 36])**.** *There exists a procedure* 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.*

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

Lemma 4.12 (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 $z_1^{\lceil \log n \rceil} \cdot \prod_{p \in \mathrm{P}\colon p < z_2} p^{\lceil \log n \rceil}$ is a non-zero multiple of $\phi(n)$.*

There exists a procedure Factor *that on input (n, z_1, z_2), runs in time $\mathrm{poly}(z_2) \cdot \mathrm{poly}(\log |nz_1|)$, 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 [25]

Theorem 4.13 (Miller's algorithm [25, 36])**.** *There exists a procedure that on input $n \in \mathbb{N}$ and $\mu \in \mathbb{Z}$, runs in time $\mathrm{poly}(\log |n\mu|)$, and if μ is a non-zero multiple of $\phi(n)$, it factors n with constant probability.*

By definition $\mu = z_1^{\lceil \log n \rceil} \cdot \prod_{p \in \mathrm{P}\colon p < z_2} p^{\lceil \log n \rceil}$ is a non-zero multiple of $\phi(n)$. Thus, Miller's algorithm on input (n, μ), 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 μ is easily computable in time $\mathrm{poly}(z_2, \log n)$. □

We are now finally ready to define Emul.

Algorithm 4.14 (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 Forger_G, return the following value to C:

1. *If 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 $n \le |C|^{\frac{\log |C|}{\log \log |C|}}$, 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 4.15.
3. *If factoring of n is successful, return $h^G(0)^d \quad [G_n]$, where $d = e^{-1} \mod \phi(n)$.*

   Otherwise, abort.

. . . .

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

Definition 4.15 (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, Forger_G} to 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 4.7, where H is as in 4.8 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 $\mathrm{rank}_e(M^H_{\mathcal{I}}) = t$.*[11]

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

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

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 Forger, the latter would not return $\perp$, but either (1) Sef failed, or (2) z was a factoring advice but Factor failed, or (3) z was not a factoring advice for n. As the first two cases happen with negligible probability (by Theorem 4.11 and Lemma 4.12), 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") is the topic of Section 5.

Lemma 4.16. *A query $q = (n, \cdot)$ to* 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 4.15.*

The following holds for any oracle-aided circuit C:

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

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

Apart from its role in the proof of Lemma 4.3 (see Section 4.3), Lemma 4.16 immediately reproves the following fact.

Corollary 4.17. *In the generic group model, the* RSA *assumption is equivalent to (the hardness of) factoring.*

Proof. We only sketch the proof. Clearly, if we can factor the integer n output by Gen (see Definition 3.8) with non-negligible probability, we can break the RSA assumption in the generic group model (by computing d such that $e^{-1} \equiv d \mod \phi(n)$). For the other direction, consider the oracle-aided algorithm A (with oracle access only to $G \in \mathcal{G}$) that on input n, sets C to be the oracle-aided circuit that makes the single call $\mathsf{Forger}(n, e, h, \{\sigma_i = i^d \ [G_n]\}_{i\in[|h|]})$, where e a random prime, h the function such that $h^G(i) = i$ and $d = e^{-1} \mod \phi(n)$. We assumed that $i \in [t]$ is relatively prime to n, otherwise we can easily factor by computing the greatest common divisor of i and n. Moreover, we assumed that e does not divide $\phi(n)$ and thus d exists. However, because e is random the latter happens with high probability. Note that C is efficient under the assumption that RSA is easy. Finally, A applies the method of Emul to emulate C^G. It is easy to see that Forger_G does not return $\perp$ on this call (since $\mathrm{rank}_e M^{G,n,H}(\{h(i)\}_{i\in[t]}) = t$, see 4.8). Hence, Lemma 4.16 yields that Emul aborts with negligible probability, and therefore A factors n with save but negligible probability. $\square$

4.3 Putting it Together

Proof of Lemma 4.3. Claim 4.10 yields that F^{G,Forger_G} breaks the security of Σ^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 $|C_k| \in \text{poly}(k)$, and that we assume without loss of generality that $|C_k| \geq k$. Theorem 4.11 and Lemma 4.12 yield that $\mathsf{Emul}(C_k)$ answers all "expected" queries of C_k to Forger with probability $1 - |C_k| \cdot 2^{-\Omega(k)} = 1 - \text{neg}(k)$, where Lemma 4.16 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. □

5 Proving Lemma 4.16 via the Short Description Paradigm

To prove Lemma 4.16 we apply the Gennaro and Trevisan [14] "short description" paradigm (introduced in the realm of black-box reduction from one-way functions/permutations) in the Generic Group Model. Roughly speaking, [14] shows that if there exists a circuit C that inverts a random permutation $\pi \in \Pi_n$ too well, then the description size of such random permutation is noticeably below $\log(n!)$ bits. This derives a contradiction, since describing a random permutation of Π_n requires $\log(n!)$ bits.

Analogously, assume towards contradiction that there exists a circuit C violating Lemma 4.16. In a nutshell, we prove that if C finds an unexpected query $q = (n, \cdot)$ for some $G \in \mathcal{G}$, then C's queries reveal a system of equations that gives a lot of information about the permutation $\pi \in \Pi_{\phi(n)}$ that defines G_n. This yields that there exists a large set of permutations inside $\Pi_{\phi(n)}$, whose members can be described using a (much) shorter than the information theoretical bound, deriving a contradiction.

In the following we assume for simplicity that C^{G,Forger_G} makes a single call to Forger_G and then halts (this simplification is easily justified in Section 5.8), and denote this query by q_G. Since C can only ask a query $q = (n, \cdot)$, an averaging argument yields that there exists $n > |C|^{\frac{\log|C|}{\log\log|C|}}$ such that

$$\mathsf{Pr}_{G \leftarrow \mathcal{G}}[\text{Break}(G, n)] > \delta(|C|)/\,|C|\,, \tag{3}$$

where predicate $\text{Break}(G, n)$ is true iff $q_G = (n, \cdot)$ and $(\det(Q_{C,G,q}), |C|^4)$ is not a factoring advice for n. To see this average argument let $(N_1, \ldots, N_{|C|})$ denote the random variables of all the distinct integers such that $q_G = (N_i, \cdot)$ is a query of C to G and N_i is the i-th distinct such integer in order of appearance. A simple argument derives that there exists an index $i^* \in [|C|]$ such that $\mathsf{Pr}_{G \leftarrow \mathcal{G}}[\text{Break}(G, N_{i^*})] > \delta(|C|)/\,|C|$, as we can assume without loss of generality that C has always made a query to G of the form $q_G = (n, \cdot)$ before querying Forger on $q = (n, \cdot)$. Moreover, notice that N_1 is fixed to some n_1, because everything is deterministic until the first query. Therefore, we can fix G_{n_1} such that $\mathsf{Pr}_{G \leftarrow \mathcal{G}}[\text{Break}(G, N_{i^*}) | G_{n_1} \in G] > \delta(|C|)/\,|C|$. Having fixed G_{n_1}, N_2 is fixed to some n_2 and we can similarly fix G_{n_2} and we can continue like that until fixing n_{i^*} and still have $\mathsf{Pr}_{G \leftarrow \mathcal{G}}[\text{Break}(G, N_{i^*}) | G_{n_1}, \ldots, G_{n_{i*-1}} \in G] > \delta(|C|)/\,|C|$. Finally, we can fix $G_{n'}$ for every $n' \neq n_{i^*}$.

Given $G = \{G_i\}_{i \in \mathbb{N}} \in \mathcal{G}$ and $\pi \in \Pi_{\phi(n)}$, where $n = n_{i*}$, let G^π be the family of groups derived from G by replacing G_n with $\pi(\mathbb{Z}_n^*)$. The above yields that there exists $G \in \mathcal{G}$ such that

$$\mathsf{Pr}_{\pi \leftarrow \Pi_{\phi(n)}}[\text{Break}(G^\pi, n)] > \delta(|C|)/\,|C| \tag{4}$$

In the following we fix such G, let $\Pi'_{\phi(n)} \subseteq \Pi_{\phi(n)}$ be the set of permutations for which $\mathrm{Break}(G^\pi, n) = 1$ and let $q^\pi := q_{G^\pi}$. Equation (4) yields that

$$\log\left|\Pi'_{\phi(n)}\right| \geq \log(\phi(n)!) + \log(\delta(|C|)/\,|C|) - 1 \tag{5}$$

We conclude the proof of Lemma 4.16 by showing how to use the assumed C for giving a much shorter description of $\Pi'_{\phi(n)}$, deriving a contradiction.

We first describe a rather simple attempt to give a short description of $\Pi'_{\phi(n)}$. While this attempt falls too short, it well illustrates the main ideas and the difficulties such a task involves. In Section 5.4 we refine this approach to get the actual short description of $\Pi'_{\phi(n)}$.

5.1 Short description of $\Pi'_{\phi(n)}$ — First attempt

Let us start with few definitions. Let H be as in 4.8 (with respect to to $q = q^\pi$) and let $(H;C)$ be as in Definition 4.15, while omitting the (single) call of C to Forger_G (hence, $(H;C)$ only assumes oracle access to G).

For $\pi \in \Pi'_{\phi(n)}$ with $q^\pi = (n, e, h, \{\sigma_i\}_{i\in[t]})$, let $\mathrm{HardWired}^\pi := \mathrm{HardWired}_{(H;C),G^\pi,n}$, $\mathrm{Terms}^\pi := \mathrm{Terms}_{(H;C),G^\pi,n}$ and $\mathrm{AnsTerms}^\pi = \mathrm{Terms}^\pi \setminus \mathrm{HardWired}^\pi$. In the following definition we identify a set of t "independent" terms inside Terms^π.

Definition 5.1 (independent terms). *For $\pi \in \Pi'_{\phi(n)}$, let $\{v_i\}_{i\in[\ell]} = \mathrm{HardWired}^\pi$, let $M = M^{G^\pi,n,H}(\{h^{G^\pi}(i)\}_{i\in[t]})$ be as in Definition 4.7, and let $\mathcal{I} \subseteq [\ell]$ be the first subset of size t such that $\mathrm{rank}_e(M_\mathcal{I}) = t$ as in Definition 4.15. We let $\mathrm{IndepHardW}^\pi$ and $\overline{\mathrm{IndepHardW}}^\pi$ be the ordered sets $\{w_i\}_{i=1}^t = \{v_i\}_{i\in\mathcal{I}}$ and $\{\overline{w}_i\}_{i=1}^{\ell-t} = \{v_i\}_{i\notin\mathcal{I}}$ respectively.*

Fix $\pi \in \Pi'_{\phi(n)}$, and let $q^\pi = (n, e, h, \{\sigma_i\}_{i\in[t]})$. Recall that we assume without loss of generality that before making the call q^π, the circuit C evaluates x_i^e (over G_n^π) for every $i \in [t]$. Let $\mathrm{Can}^\pi(\cdot) := \mathrm{Can}_{(H;C),G^\pi,n}(\cdot)$, let $\{w_i\}_{i=1}^t = \mathrm{IndepHardW}^\pi$, and let $\{\overline{w}_i\}_{i=1}^{\ell-t} = \overline{\mathrm{IndepHardW}}^\pi$. For $i \in [t]$, assume that $\mathrm{Can}^\pi(h(i)) = \prod_{j=1}^{\ell-t} \overline{w}_i^{\beta_{ij}} \cdot \prod_{j=1}^t w_j^{\alpha_{ij}}$ [Symb] and that $\mathrm{Can}^\pi(\sigma_i) = \prod_{j=1}^{\ell-t} \overline{w}_j^{\beta'_{ij}} \cdot \prod_{j=1}^t w_j^{\alpha'_{ij}}$ [Symb]. Since $h(i) \equiv \sigma_i^e \;\; [\pi(\mathbb{Z}_n^*)]$ for every $i \in [t]$, it holds that

$$\prod_{j=1}^{\ell-t} \overline{w}_j^{\beta_{ij}} \cdot \prod_{j=1}^{t} w_j^{\alpha_{ij}} \equiv \prod_{j=1}^{\ell-t} \overline{w}_i^{e\cdot\beta'_{ij}} \cdot \prod_{j=1}^{t} w_j^{e\cdot\alpha'_{ij}} \quad [\pi(\mathbb{Z}_n^*)]. \tag{6}$$

Or put it differently,

$$\prod_{j=1}^{t} w_j^{\alpha_{ij}-e\cdot\alpha'_{ij}} \equiv \prod_{j=1}^{\ell-t} \overline{w}_j^{e\cdot\beta'_{ij}-\beta_{ij}} \quad [\pi(\mathbb{Z}_n^*)],$$

which yields the following set of equations over $\mathbb{Z}_n^*$:

$$\left\{\prod_{j=1}^{t} \pi^{-1}(w_j)^{\alpha_{ij}-e\cdot\alpha'_{ij}} \equiv \prod_{j=1}^{\ell-t} \pi^{-1}(\overline{w}_j)^{e\cdot\beta'_{ij}-\beta_{ij}} \quad [\mathbb{Z}_n^*]\right\}_{i\in[t]}. \tag{7}$$

We denote the above set of t equation by E^π, and define their "solution set" as follows:

Definition 5.2 (solution set)**.** *Given a set of equations* E *over* $\mathbb{Z}_n^*$*, we let* $\mathcal{S}(\mathrm{E})$ *denote the (solution) set of all* t*-tuples over* $\mathbb{Z}_n^*$*, for which the equations of* E *still hold when the value of* $(\pi^{-1}(w_1),\ldots,\pi^{-1}(w_t))$ *in the equations is replaced with any (but same for all equations) element of* $\mathcal{S}(\mathrm{E})$*.*

The following lemma helps us yield that the assumption that $(\det(Q_{C,G^\pi,q}),|C|^4)$ is not a factoring advice for n, implies that $\mathcal{S}(\mathrm{E}^\pi)$ is rather small. Namely, E^π gives a lot of information about the preimages of $\mathrm{IndepHardW}^\pi$ (i.e., much smaller than the upper bound of $\phi(n)^t$).

Lemma 5.3. *Let* $A=\{a_{ij}\}_{i,j\in[t]}$ *be an integer matrix, let* $\{b_i\in\mathbb{Z}_n^*\}_{i\in[t]}$*, and let* E *be the following set of equations:*

$$\left\{\prod_{j\in[t]} x_j^{a_{ij}} \equiv b_i \quad [\mathbb{Z}_n^*]\right\}_{i\in[t]}.$$

Assuming that $\det A\neq 0$*, then for every* $c\in\mathbb{N}$ *such that* $(\det A,c)$ *is* not *a factoring advice for* n *(see Lemma 4.12), there exists a prime* $r\geq c$ *such that* $r|\phi(n)$ *and* $|\mathcal{S}(\mathrm{E})|\leq(\frac{\phi(n)}{r})^t$*.*

The proof of Lemma 5.3 is given in Section 5.9. However, the intuition is that an equation (consider the simple case of one unknown) of the form $x^a\equiv b\ [\mathbb{Z}_n^*]$ has "few" solutions, iff $\gcd(a,\phi(n))$ is small. In our case, remember that matrix Q_{C,G^π,q^π} corresponds to the coefficients of the exponents in E^π and the fact that $(\det(Q_{C,G_\pi,q}),c)$ is not a factoring advice implies that $\gcd(\det(Q_{C,G_\pi,q}),\phi(n))$ is "small". For our needs, Lemma 5.3 yields the following upper bound on $|\mathcal{S}(\mathrm{E}^\pi)|$ (also proven in Section 5.9):

Claim 5.4. *It holds that* $|\mathcal{S}(\mathrm{E}^\pi)|\leq(\frac{\phi(n)}{|C|^4})^t$*.*

The following description tries to exploit the above fact to give a too short description for $\Pi'_{\phi(n)}$. It contains the information needed to emulate $(H;C)^{G^\pi}$, the preimages (with respect to π) of all terms in $\overline{\mathrm{IndepHardW}}^\pi$, and the information needed to describe π outside of Terms^π. Since given the emulation and $\pi^{-1}(\overline{\mathrm{IndepHardW}}^\pi)$, it is possible to reconstruct E^π, Claim 5.4 yields that relatively little information is needed to compute the preimages of all terms in $\mathrm{IndepHardW}^\pi$. Using the above, the preimages of all terms in Terms^π can be computed, and thus (using the final part of the description) all of π can be reconstructed.

Description 5.5 (description of $\pi\in\Pi'_{\phi(n)}$)**.**
Let $q^\pi=(n,e,h,\{\sigma_i\}_{i\in[t]})$*. The description consists of the following parts defined with respect to the execution of* $(H;C)^{G^\pi}$ *as follows.*

1. *Description of* h*.*
2. *Ordered set* $\pi^{-1}(\overline{\mathrm{IndepHardW}}^\pi)$*, given in appearance order.*
3. *Ordered set* AnswFull *of all the answer terms, given in appearance order.*
4. *An index* SolutionInd *to the solution of* $\mathcal{S}(\mathrm{E}^\pi)$ *induced by* π*.*
5. *A description* Rest *of the values of* π *over* $\mathbb{Z}_n^*\setminus\pi^{-1}(\mathrm{Terms}^\pi)$*.*

We first show how to use the above description to reconstruct the value of π (yielding that this is indeed a good description), and then try to upper bound its length.

5.2 Reconstructing π

Lemma 5.6. *For any $\pi \in \Pi'_{\phi(n)}$, the description of π given in Description 5.5 (together with the fixed values of $G \in \mathcal{G}$ and the assumed circuit C) determines the value of π.*

Proof. Recall that saved but the description of π, we are given the value of the assumed circuit C and the value of the *fixed* set $G = (G_1, \ldots, G_{n'}) \in \mathcal{G}$, where n' is the maximal value accessed by $(H; C)$ (here we abuse notation because actually G should be an infinite set). We use the following reconstructing algorithm:

Description 5.7 (reconstruction)**.**

1. *Emulate the execution of $(H; C)^{G^\pi}$ as follows:*
   (a) *$G^\pi_{n'}$-queries with $n' \neq n$ are answered using G.*
   (b) *G^π_n-queries are answered using* AnswFull*.*
2. *Use the above emulation to compute π as follows:*
   (a) *Identify (using the emulation) the canonical forms of the terms of $(H; C)$ with respect to G^π and n, and use it to identify of the ordered sets* $\mathrm{IndepHardW}^\pi$ *and* $\overline{\mathrm{IndepHardW}}^\pi$ *and the equations set* E^π*.*
   (b) *Identify the solution set $\mathcal{S}(\mathrm{E}^\pi)$ and use* SolutionInd *to determine $\pi^{-1}(w)$ for every $w \in$* $\mathrm{IndepHardW}^\pi$).
   (c) *Use the canonical form of* Terms^π*, and the values of* $\pi^{-1}(\mathrm{HardWired}^\pi)$ *(determined by* $\pi^{-1}(\overline{\mathrm{IndepHardW}}^\pi)$ *and* $\pi^{-1}(\mathrm{IndepHardW}^\pi)$*) to compute* $\pi^{-1}(\mathrm{Terms}^\pi \setminus \mathrm{HardWired})$.
   (d) *Use* Rest *to fully reconstruct π.*

....

The only non-trivial step is Step 2(c). Recall that the canonical form c of a term $u \in \mathrm{Terms}^\pi$, with respect to G^π and n, represents u as a multiplication of hardwired terms. Namely

$$c \equiv \prod_{i \in [\ell]} v_i^{a_i} \quad [\mathrm{Symb}],$$

where $\{v_i\}_{i \in [\ell]} = \mathrm{HardWired}^\pi$. It follows that $u \equiv \prod_{i \in [\ell]} v_i^{a_i} \quad [\pi(\mathbb{Z}_n^*)]$, and therefore

$$\pi^{-1}(u) \equiv \prod_{i \in [\ell]} \pi^{-1}(v_i)^{a_i} \quad [\mathbb{Z}_n^*].$$

Hence, $\pi^{-1}(u)$ is determined by $\pi^{-1}(\mathrm{HardWired}^\pi)$. ∎

5.3 Description length

The value of $\pi^{-1}(\overline{\mathrm{IndepHardW}}^\pi)$ can be described using about $(\ell - t)\log\phi(n)$ bits (actually $\log\frac{\phi(n)!}{(\phi(n)-\ell+t)!} + 1$ bits suffice). AnswFull can be described using about $|\mathrm{AnswFull}|\log\phi(n)$ bits. In addition, given Terms^π and $\pi^{-1}(\text{Terms}^\pi)$, Rest can be described using about $\log(\phi(n) - |\text{Terms}^\pi|)! \approx (\phi(n) - |\text{Terms}^\pi|) \log \phi(n)$ bits. It follows that

$$\begin{aligned}\text{DescSize}(\pi) &\approx |h| + (|\text{AnswFull}| + \ell - t) \cdot \log \phi(n) + |\text{SolutionInd}| + (\phi(n) - |\text{Terms}^\pi|) \log \phi(n) \\ &\approx (\phi(n) - |\text{Terms}^\pi| + |\text{AnswFull}| + \ell - t) \cdot \log \phi(n) \\ &\quad + t \cdot (\log \phi(n) - 4 \log |C|),\end{aligned}$$

where the second inequality follows since $|h| \leq t$ and Claim 5.4.

Assuming that AnswFull does not contain hardwired terms or reoccurrences of the same value, it follows that $|\text{AnswFull}| = |\text{Terms}^\pi| - \ell$. Therefore,

$$\begin{aligned}\text{DescSize}(\pi) &\approx (\phi(n) - t) \cdot \log \phi(n) + t \cdot \log \phi(n) - 4t \cdot \log |C| \\ &\approx \phi(n) \cdot \log \phi(n) - 4t \cdot \log |C| \\ &\leq \phi(n) \cdot \log \phi(n) - \Theta(\frac{\log^3 |C|}{\log \log |C|}).\end{aligned}$$

Where to derive the inequality we observed that $t \geq |h| \geq \log n$ otherwise hash function h could not have as input elements of $\mathbb{Z}_n^*$ and by assumption $\log n \geq \frac{\log^2 |C|}{\log \log |C|}$. Comparing the above to Equation (5), it follows that we the description of $\Pi'_{\phi(n)}$ is $\Omega(\log |C|)$ bits too short.

Unfortunately, the assumption about AnswFull we made above is not necessarily true. Firstly, AnswFull might contain values in HardWired^π, and secondly, there might be two (or more) queries in the execution of $(H; C)^{G^\pi}$ that yield the same output. Such "collisions" might inflate the size of AnswFull by at least $|C|$ bits, causing the above description to be too long.[12]

Let us elaborate on the above issue. Simple collisions that inflates the size of AnswFull are identical queries. This case, however, is easy to handle, since the identical queries are easily tractable and we only keep their answer in their first appearance in AnswFull. More generally, whenever the response of G_n^π is information theoretically implied by the description of π and the emulation till this point, we can omit it from the set AnswFull. Some collisions, however, are not information theoretically determined. For instance, the execution of $(H; C)^{G^\pi}$ might start with the following two queries:

$$G_n^\pi(w_1, w_2) \tag{8}$$
$$G_n^\pi(w_3, w_4) \tag{9}$$

where w_1, w_2, w_3, w_4 are different terms inside IndepHardW^π. Generally, for some permutations in $\Pi'_{\phi(n)}$ these queries collide (have the same answer), where this is not the case for other permutations. The point is that the output of the second query is *not* information theoretically determined by the output of the first query and the above description of π.

Remark 5.8 (collisions)**.** *Somewhat ironically, collisions in the execution of $(H; C)^{G^\pi}$ (in particular those reflected in Equation (6)) are exactly what gave the hope that we can use C to compress $\Pi'_{\phi(n)}$. In the following section, we refine the above approach (and in particular the way collisions are treated) to get a description that is guaranteed to be short enough.*

[12] Note that such penalty in the description length is payed, even if we only describe the second query in the collision via an index for the first query.

5.4 Short description of $\Pi'_{\phi(n)}$ — The actual approach

The approach we used above for describing the execution of $(H;C)^{G^\pi}$ was too wasteful to be compensable by what we save on describing the preimages of $\mathrm{IndepHardW}^\pi$. In the following we refine the way this information is described, yielding an overall shorter description size.

Collision queries, informally defined in the previous section, play an important role in the following discussion.

Definition 5.9 (collisions)**.** *A G_n^π-query made through the execution of $(H;C)^{G^\pi}$ is a* collision*, if its answer term has already appeared in the evaluation order in another G_n^π-query either as a hardwired term or as the answer term to a previous query.*

We start by considering the equations induced by a prefix of the queries made by $(H;C)^{G^\pi}$. Generalizing Equation (7), the j'th G_n^π-query made by $(H;C)^{G^\pi}$ yields either some equalities or $j-1$ inequalities over $\mathbb{Z}_n$. That is, let u_j be the j'th answer term in Terms^π, and let $\prod_{j=1}^{\ell-t} \overline{w}_i^{\beta_{ij}} \cdot \prod_{j=1}^{t} w_j^{\alpha_{ij}}$ be its canonical form, where $\{w_i\}_{i=1}^t = \mathrm{IndepHardW}^\pi$, and let $\{\overline{w}_i\}_{i=1}^{\ell-t} = \overline{\mathrm{IndepHardW}^\pi}$. If the answer term u_j induces a collision with some term u_k, then this equality yields the equation

$$\prod_{i=1}^{t} \pi^{-1}(w_i)^{\alpha_{ij}-\alpha_{ik}} \equiv \prod_{i=1}^{\ell-t} \pi^{-1}(\overline{w}_i)^{\beta_{ik}-\beta_{ij}} \quad [\mathbb{Z}_n^*]. \tag{10}$$

Otherwise (no collision), it yields the following inequality for any $k < j$

$$\prod_{i=1}^{t} \pi^{-1}(w_i)^{\alpha_{ij}-\alpha_{ik}} \not\equiv \prod_{i=1}^{\ell-t} \pi^{-1}(\overline{w}_i)^{\beta_{ik}-\beta_{ij}} \quad [\mathbb{Z}_n^*]. \tag{11}$$

We denote by E_j^π the set of equations/inequalities induced by the first j queries of $(H;C)^{G^\pi}$. To avoid notational clattering, we assume that the j'th answer term is also the j'th term in Terms^π. It is also not hard to see that the value of $\mathcal{S}(\mathrm{E}_j^\pi)$, and the execution of the first $j-1$ steps, determines the value of $k < j$ such that $u_j = u_k$, if such exists.

In the following description of π does not automatically add the answer term u_j to AnswFull (as done in the first attempt), but rather adds to the description the value γ_j that describes the solution set $\mathcal{S}(\mathrm{E}_j^\pi)$ using $\mathcal{S}(\mathrm{E}_{j-1}^\pi)$. Only if j'th query is non colliding (as reflected in the description of $\mathcal{S}(\mathrm{E}_j^\pi)$), the value of u_j is added to the set AnswFull.

Observe that the above description together with the value of h, $\pi^{-1}(\overline{\mathrm{IndepHardW}^\pi})$ and Rest (see Description 5.5), fully determines the value of π — In the emulation of $(H;C)^{G^\pi}$ one can use the above information to decide whether the j'th query induces a collision or not (and with whom). If positive, it answers the query using the value of its colliding term. Otherwise, it answers the query using the next value in AnswFull.

The above description is also not wasteful; any information given in the description, gives (essentially) the same amount of info about π. To shorten the above description further (so that it is shorter than it should be), we use the following claim (proof given in Section 5.9.4), that is in the spirit (and indeed, implied by the same lemma) of the main observation made in our first attempt (Claim 5.4).

Claim 5.10 (informative indices)**.** *There exists a set of indices* $\mathrm{InformativeIndx}_\pi = \{i_1, \ldots, i_t\} \subseteq [|(H;C)|]$ *of collision queries inside the execution of* $(H;C)$*, such that the following holds:*

$$\prod_{j=1}^{t} \frac{|\mathcal{S}(\mathrm{E}^\pi_{i_j-1})|}{|\mathcal{S}(\mathrm{E}^\pi_{i_j})|} \geq (\frac{1}{2} \cdot |C|^4)^t.$$

We modify the above description as follows: we add the description of $\mathrm{InformativeIndx}_\pi$, and for each $j \in \mathrm{InformativeIndx}_\pi$ remove the value γ_j (describing the solution set $\mathcal{S}(\mathrm{E}^\pi_j)$ using $\mathcal{S}(\mathrm{E}^\pi_{j-1})$), and add the index of the colliding term u_k. All in all, we roughly add additional $2t \cdot \log|(H;C)|$ bits, and remove $4t\log|C|$. Since $|(H;C)| \leq 2\,|C|$, we save $\Omega(t\log|C|)$ from the description size, making the overall description too short.

The actual description of π is somewhat more complicated. In particular, this happens since the amount of information each γ_j contains might be fractional, and we cannot allow ourselves the price of rounding each γ_j. The solution we take is to bundle together the γ_j whose indices fall between two informative indices ($O(t)$ such bundles) and thus waste only t bits for rounding.

Description 5.11 (short description of $\pi \in \Pi'_{\phi(n)}$)**.**
Let $q^\pi = (n, e, h, \{\sigma_i\}_{i\in[t]})$*. The description consists of the following parts defined with respect to the execution of* $(H;C)^{G^\pi}$ *as follows.*

1. *Description of* h*.*
2. *Ordered set* AnswFull *of all non-colliding answer terms, in appearance order.*
3. *Ordered set* $\pi^{-1}(\overline{\mathrm{IndepHardW}}^\pi)$*, in appearance order.*
4. *Ordered index set* IndepHardWIndx*, describes where the terms in* $\mathrm{IndepHardW}^\pi$ *appear for the first time.*
5. *Description of the set* $\mathrm{InformativeIndx}_\pi = \{j_1, \ldots, j_t\}$*, defined in Claim 5.10.*
6. *Order index set* CollisionIndx*, contains the terms that collide with the answer items of* $\mathrm{InformativeIndx}_\pi$*.*
7. *Ordered set* $\{\gamma_1, \ldots, \gamma_t\}$*, where* γ_i *describes for* $j_i \in \mathrm{InformativeIndx}_\pi$*, the set* $\mathcal{S}(\mathrm{E}_{j_i-1})$ *using* $\mathcal{S}(\mathrm{E}_{j_{i-1}})$ *(taking* $\mathcal{S}(\mathrm{E}_{j_0}) = \mathbb{Z}_n^{*\,t}$*).*
8. *An index* SolutionInd *to the solution of* $\mathcal{S}(\mathrm{E}_{j_t})$ *induced by* π*.*
9. *A description* Rest *of the values of* π *over* $\mathbb{Z}_n^* \setminus \pi^{-1}(\mathrm{Terms}^\pi)$*.*

The above description does not specify the way the set $\mathcal{S}(\mathrm{E}_{j_i-1})$ is described using $\mathcal{S}(\mathrm{E}_{j_{i-1}})$, and it is still unclear why the above description determines the value of π. We prove that in Sections 5.6 and 5.7, but first let us compute the length of the above description, under the assumption (also proven in Section 5.7) that it takes $\left\lceil \log \frac{|\mathcal{S}(\mathrm{E}_{j_{i-1}})|}{|\mathcal{S}(\mathrm{E}_{j_i-1})|} \right\rceil + 1$ bits to describe γ_i.

5.5 Description length

Let us see how many bits are required at most to describe each part of the above description (the i-th item upper bounds the length of the i-th item above):

1. t
2. $\lceil \log \phi(n)^{|\text{AnswFull}|} \rceil + 1$
3. $(\ell - t) \cdot \log \phi(n) + 1$
4. $t \cdot \lceil \log 2|C| \rceil$ (since $|(H;C)| \leq 2\,|C|$)
5. $t \cdot \lceil \log 2|C| \rceil$
6. $t \cdot \lceil \log 2|C| \rceil$
7. $\sum_{j=1}^{t} (\log \frac{|\mathcal{S}(\mathrm{E}_{j_{i-1}})|}{|\mathcal{S}(\mathrm{E}_{j_i-1})|} + 1)$
8. $\log |\mathcal{S}(\mathrm{E}_{i_t})| + 1$
9. $\log\,(\phi(n) - |\text{AnswFull}| - \ell)! + 1$

Notice that

$$\begin{aligned}(7) + (8) =& \sum_{j=1}^{t} (\log \frac{|\mathcal{S}(\mathrm{E}_{j_{i-1}})|}{|\mathcal{S}(\mathrm{E}_{j_i-1})|} + 1) + \log |\mathcal{S}(\mathrm{E}_{i_t})| + 1 \\ =& \sum_{j=1}^{t} (\log \frac{|\mathcal{S}(\mathrm{E}_{j_{i-1}})|}{|\mathcal{S}(\mathrm{E}_{j_i-1})|} + \log \frac{|\mathcal{S}(\mathrm{E}_{j_i-1})|}{|\mathcal{S}(\mathrm{E}_{j_i})|}) + \log |\mathcal{S}(\mathrm{E}_{i_t})| \\ & - \sum_{j=1}^{t} \log \frac{|\mathcal{S}(\mathrm{E}_{j_i-1})|}{|\mathcal{S}(\mathrm{E}_{j_i})|} + t + 1.\end{aligned}$$

Since $\mathcal{S}(\mathrm{E}_{ij_0}) = \mathbb{Z}_n^{*\,t}$, by Claim 5.10 we have

$$(7) + (8) = t \cdot \log \phi(n) - 4t \cdot \log |C| + O(t).$$

It follows that

$$\begin{aligned}(2) + (3) + (7) + (8) =& (|\text{AnswFull}| + \ell) \cdot \log \phi(n) - 4t \cdot \log |C| + O(t) \\ =& \log \phi(n)^{|\text{AnswFull}| + \ell} - 4t \cdot \log |C| + O(t)\end{aligned}$$

Since

$$(1) + (4) + (5) + (6) = 3t \cdot \log |C| + O(t),$$

we conclude that

$$\text{DescSize}(\pi) = \log\,(\phi(n) - |\text{AnswFull}| - \ell)! + \log \phi(n)^{|\text{AnswFull}| + \ell} - t \cdot \log |C| + O(t) \tag{12}$$

By Equation (5) $\log \left|\Pi'_{\phi(n)}\right| \geq \log(\phi(n)!) + \log(\delta(|C|)/|C|) - 1$. Hence, for every $\pi \in \Pi'_{\phi(n)}$ it holds that

$$\begin{aligned}
&\log \left|\Pi'_{\phi(n)}\right| - \mathrm{DescSize}(\pi) \\
&\geq \log(\phi(n)!) + \log(\delta(|C|)/|C|) - 1 - \log\left(\phi(n) - |\mathrm{AnswFull}| - \ell\right)! - \log \phi(n)^{(|\mathrm{AnswFull}|+\ell)} \\
&\quad + t \cdot \log|C| - O(t) \\
&\geq \log\left(\phi(n) - |\mathrm{AnswFull}| - \ell\right)^{|\mathrm{AnswFull}|+\ell} - \log \phi(n)^{(|\mathrm{AnswFull}|+\ell)} + \log(\delta(|C|)/|C|) \\
&\quad + t \cdot \log|C| - O(t).
\end{aligned}$$

Noticing that

$$\begin{aligned}
\log\left(\phi(n) - |\mathrm{AnswFull}| - \ell\right)^{|\mathrm{AnswFull}|+\ell} - \log \phi(n)^{|\mathrm{AnswFull}|+\ell} &= \log\left(1 - \frac{|\mathrm{AnswFull}| + \ell}{\phi(n)}\right)^{|\mathrm{AnswFull}|+\ell} \\
&\geq \log\left(1 - \frac{(|\mathrm{AnswFull}| + \ell)^2}{\phi(n)}\right) \\
&= -O(1),
\end{aligned}$$

and that $t \geq \log n \geq \frac{\log^2|C|}{\log\log|C|}$ (as we argued in the previous section), we have that

$$\begin{aligned}
\log \left|\Pi'_{\phi(n)}\right| - \mathrm{DescSize}(\pi) &\geq -\log^2|C| - \log|C| + \Theta\left(\frac{\log^3|C|}{\log\log|C|}\right) \\
&= \Omega(\log|C|),
\end{aligned}$$

Namely, our description of π is $\Omega(\log|C|)$ bits too short. In the next section prove that the description given in 5.11 uniquely determines π. Thus would yield a too short description of a permutation in $\Pi'_{\phi(n)}$, and thus would prove Lemma 4.16.

5.6 Reconstructing π

In this section we show how to use the description of π given in Description 5.11 to fully reconstruct π. Namely, we prove the following lemma:

Lemma 5.12. *For any $\pi \in \Pi'_{\phi(n)}$, the description of π given in Description 5.11 (together with the fixed values of $G \in \mathcal{G}$ and the assumed circuit C) determines the value of π.*

Proof. We use the following claim:

Claim 5.13. *For $\pi \in \Pi'_{\phi(n)}$, assume that the description of h,* AnswFull*, $\pi^{-1}(\overline{\mathrm{IndepHardW}^{\pi}})$ and* IndepHardWIndx*, as described in Description 5.11 are known. Then for any $j \in [|(H;C)|]$, the description of the set $\mathcal{S}(\mathrm{E}_j)$ determines the first j steps of the execution of $(H;C)^{G^\pi}$, and vice versa.*

Proof. Assume that the claim holds for $j-1$, we use description of the first $j-1$ steps of $(H;C)^{G^\pi}$, to emulate its j'th step as follows: using the description of circuit $(H;C)$ and the description of the first $j-1$ steps, we determine the j'th query q_j. If q_j is a $G^{\pi}_{n'}$-query with $n' \neq n$, we answer it using the (fixed) description of G, otherwise we do the following:

Let a_i be the answer of the i-query of $(H;C)^{G^{\pi}}$, and let c_i the value of the first $k < i$ such that $a_i = a_k$ (set it to $\perp$ if no such k exists). The main observation (that we prove below) is that given the execution of the first $j-1$ steps, the value of $\mathcal{S}(\mathrm{E}_j)$ determines the value of c_j and vice versa. It follows that the first j steps determine $\mathcal{S}(\mathrm{E}_j)$, where for the other direction of the lemma, we answer the j'th query as follows: we first compute c_j, and then answer with a_{c_j} if $c_j \neq \perp$, and with the next value of IndepHardWIndx otherwise.

For proving the above observation, we first notice that the value of E_{j-1} is determined by $\pi^{-1}(\overline{\mathrm{IndepHardW}^{\pi}})$ and IndepHardWIndx, and the execution of the first $j-1$ steps. Similarly, the value E_j is determined by the value c_j (and the above values). By the definition of E_j, different values for c_j would imply different value for E_j. Moreover, such two different values for E_j cannot hold simultaneously — By definition, $c_j = k \in [j-1]$ yields

$$\prod_{i=1}^{t} \pi^{-1}(w_i)^{\alpha_{ij}-\alpha_{ik}} \equiv \prod_{i=1}^{\ell-t} \pi^{-1}(\overline{w}_i)^{\beta_{ik}-\beta_{ij}} \quad [\mathbb{Z}_n^*],$$

but since a_k is the first collision, $c_j = k$ also yields that

$$\prod_{i=1}^{t} \pi^{-1}(w_i)^{\alpha_{ij}-\alpha_{ik'}} \not\equiv \prod_{i=1}^{\ell-t} \pi^{-1}(\overline{w}_i)^{\beta_{ik'}-\beta_{ij}} \quad [\mathbb{Z}_n^*],$$

for any $k' < k$ (where for $c_j = \perp$, the latter holds for $k = j$). Hence, the equations induced by different values of c_j conflict each other. It follows that the value of c_j induces a separation on the solution set $\mathcal{S}(\mathrm{E}_{j-1})$, and thus the value of $\mathcal{S}(\mathrm{E}_j)$ and c_j determine each other. $\square$

The above claim yields the following reconstruction algorithm for π:

Description 5.14 (reconstruction)**.**
Emulate the execution of $(H;C)^{G^{\pi}}$ through the following t steps, where the i step emulates the execution of $(H;C)^{G^{\pi}}$ between the $j_{i-1}+1$ and the j_i queries as follows:

1. *Compute $\mathcal{S}(\mathrm{E}_{j_{i-1}})$ (using the emulation so far) and use it together with the value of γ_i, to compute $\mathcal{S}(\mathrm{E}_{j_i-1})$.*

2. *Answer $G^{\pi}_{n'}$-queries with $n' \neq n$ using the (fixed) description of G, where G^{π}_{n}-queries are answered (according to their order) as follows:*

   $j < j_m$**:** *Use $\mathcal{S}(\mathrm{E}_{j_i-1})$ to determine whether this is a collision query or not. If positive, return the colliding term value. Otherwise, return the next value in* AnswFull*.*

   $j = j_m$**:** *Use* CollisionIndx *to find the colliding term, and return its value.*

The reconstruction continues as follows:

1. *Use* SolutionInd *and the value of $\mathcal{S}(\mathrm{E}_{j_t})$ to compute $\pi^{-1}(\mathrm{IndepHardW}^{\pi})$.*

2. *Use the emulation and $\pi^{-1}(\mathrm{HardWired})$ (determined by $\pi^{-1}(\overline{\mathrm{IndepHardW}^{\pi}})$ and $\pi^{-1}(\mathrm{IndepHardW}^{\pi})$) to compute $\pi^{-1}(\mathrm{Terms}^{\pi} \setminus \mathrm{HardWired})$.*

3. Use Rest *to fully reconstruct* π*.*

Claim 5.13 yields that the emulation step is guaranteed to succeed, and the rest of the correctness of the rest of the reconstruction easily follows as in the case of our first attempt (see Section 5.2). $\square$

5.7 Describing the solution sets

Lemma 5.15. *For any* $\pi \in \Pi'_{\phi(n)}$ *and* $j \in |(H;C)|$*, the execution of* $(H;C)^{G^\pi}$ *until step* j_{i-1}*, and the values of* AnswFull*,* $\pi^{-1}(\overline{\text{IndepHardW}^\pi})$*,* IndepHardWIndx *and* $\text{InformativeIndx}_\pi$*, determine a prefix free binary code* $\mathcal{C} = \{cw_k\}_{k=1}^m$*, such that the value of* $\mathcal{S}(\mathrm{E}_{j_i-1})$ *is determined by* cw_k *for some* $k \in [m]$ *(and the above values), and* $|cw_k| \leq \left\lceil \log \frac{|\mathcal{S}(\mathrm{E}_{j_{i-1}})|}{|\mathcal{S}(\mathrm{E}_{j_i-1})|} \right\rceil + 1$*.*

Proof. We use the following lemma uses an encoding called Shannon-Fano (Shannon [34]) and its proof is in Section 5.9.

Lemma 5.16. *Let* $\mathcal{Y}$ *be a finite set and let* $\{\mathcal{Y}_k\}_{k=1}^m$ *be disjoint subsets of* $\mathcal{Y}$*. Then there exists a prefix free binary code* $\mathcal{C} = \{cw_k\}_{k=1}^m$*, such that* cw_k *determines* $\mathcal{Y}_k$ *for every* $k \in [m]$*, and* $|cw_k| \leq \left\lceil \log \frac{|\mathcal{Y}|}{|\mathcal{Y}_k|} \right\rceil + 1$*.*

For some $i \in [t]$, let $\mathcal{Y} = \mathcal{S}(\mathrm{E}_{j_{i-1}})$. We will show that AnswFull, $\pi^{-1}(\overline{\text{IndepHardW}^\pi})$, IndepHardWIndx and $\text{InformativeIndx}_\pi$ define a set $\{\mathcal{Y}_k\}_{k=1}^m$ of disjoint subsets of $\mathcal{Y}$ such that for some $k \in [m]$, $\mathcal{Y}_k = \mathcal{S}(\mathrm{E}_{i_j-1})$. Then using Lemma 5.16 we prove the claim.

For each $s = (s_1, \ldots, s_t) \in \mathcal{S}(\mathrm{E}_{j_{i-1}})$, we can do the following simulation of $(H;C)$ from query j_{i-1} until $j_i - 1$: for query $M \in \{j_{i-1}, \ldots, j_i - 1\}$ of the form (u, u')

- If term u is indexed in IndepHardWIndx as the a_M-th independent hardwired term of $(H;C)^{G^\pi}$ and this is the first time this term appears in this emulation and the canonical form with respect to this emulation of term u' is $\prod_{i=1}^{a_M-1} w_i'^{\alpha_i} \cdot \prod_{i=1}^{\ell'} \overline{w}_i'^{\beta_i}$ [Symb], where w_i' is the hardwired term of this emulation indexed by the i-th index of IndepHardWIndx and $\{\overline{w}_i'\}_{i\in[\ell']}$ are all the other hardwired terms of this emulation, then we compute the 'pseudo-preimage' of the output of this query as

$$s_{a_M} \cdot \prod_{i=1}^{a_M-1} s_i^{\alpha_i} \cdot \prod_{i=1}^{\ell'} \overline{\theta}_i'^{\beta_i} \mod n,$$

  where $\overline{\theta}_i'$ is the i-th element of $\pi^{-1}(\overline{\text{IndepHardW}^\pi})$. If this value collides with the 'pseudo-preimage' of some other term u'' of this emulation (which is computed analogously), then we take u'' as the response of this query, otherwise we take as response the corresponding element of AnswFull.

- If term u is indexed in IndepHardWIndx as the a_M-th independent hardwired term of $(H;C)^{G^\pi}$ and this is not the first time this term appears in this emulation, then we stop the emulation.

- The other cases are handled similarly to the first case.

Now for every $s \in \mathcal{S}(\mathrm{E}_{j_{i-1}})$, whose simulation was not stopped, let $\mathrm{E}^s_{j_i-1}$ be the set of all equalities and inequalities of the emulation given by s with unknowns the preimages of the hardwired terms of this simulation indexed by IndepHardWIndx.

It is clear that for each $s' \in \mathcal{S}(\mathrm{E}^s_{j_i-1})$, it holds that $\mathrm{E}^s_{j_i-1} = \mathrm{E}^{s'}_{j_i-1}$, because in the emulations with respect to s and s' we will have the same collisions among the 'pseudo-preimages' and therefore these emulations are identical. Let $\{\mathcal{Y}_k\}_{k=1}^m = \{\mathcal{S}(\mathrm{E}^s_{j_i-1}) \colon s \in \mathcal{S}(\mathrm{E}_{j_{i-1}})\}$. Notice that the sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_m$ are disjoint. If $s \in \mathcal{Y}_{k_1} \cap \mathcal{Y}_{k_2}$ for distinct $k_1, k_2 \in [m]$, then we should have $\mathcal{Y}_{k_1} = \mathcal{S}(\mathrm{E}^s_{j_i-1}) = \mathcal{Y}_{k_2}$, which implies $k_1 = k_2$. Moreover, it is clear that there exists a $k \in [m]$ such that $\mathcal{Y}_k = \mathcal{S}(\mathrm{E}_{j_{i-1}})$. $\square$

5.8 Removing the simplifying assumption about C

In this section we need to justify the simplifying assumption that $C^{G,\mathsf{Forger}}$ makes only a single call to Forger and then halts. Let C be a circuit such that

$$\mathsf{Pr}_{G \leftarrow \mathcal{G}}[\mathrm{Break}(G, n)] > \delta(|C|),$$

and consider the circuit C' that before calling to Forger on $q = (n, \cdot)$, it first tries the approach that $\mathsf{Emul}(C)$ uses to emulate this call (i.e., using algorithms Sef and Factor as done in Algorithm 4.14). Specifically, since Sef and Factor are randomized algorithms, C' has hardwired the best possible random coins for these algorithms. If the factoring succeeds, C' uses it to answer the query by itself (as Emul does) and continue the execution, where otherwise it queries Forger and halts.

It is clear that C' success probability in making $\mathrm{Break}(G, n)$ true is exactly that of C. Moreover, even though C' is much larger than C, the number of calls it makes to the oracle G is at most $|C|$ more than the number of calls done by C; emulating a query to Forger, involves at most one call to G (again see Algorithm 4.14). It follows that to index a query of C' to G, we need $\log|(H; C)| + O(1)$ bits (exactly as we assumed in the simplified case!). Thus, the proof above we gave under the simplifying assumption, also holds for the general case.

5.9 Missing Proofs

5.9.1 Proof of Lemma 5.3

Let

$$E = \left\{ \prod_{j=1}^{t} x_j^{a_{ij}} \equiv d_i \mod n \right\}_{i \in [t]}$$

be a system of equations, where x_j's are the unknowns and let us see how we can bound $|\mathcal{S}(E)|$. By the Fundamental Theorem of Finite Abelian Groups, we know that if $\phi(n) = \prod_{k=1}^{s} p_i^{\alpha_k}$ is the factorization of $\phi(n)$ then $\mathbb{Z}_n^* \cong \prod_{k=1}^{s}(\prod_{l=1}^{s_k} \mathbb{Z}_{p_k^{\beta_{kl}}})$, where $\sum_{l=1}^{s_k} \beta_{kl} = \alpha_k$ and $\mathbb{Z}_{p_k^{\beta_{kl}}}$ is the additive group of integers $\mathrm{mod} p_k^{\beta_{kl}}$.

The equivalence of the groups implies a bijection between their group elements, such that each unknown x_j over $\mathbb{Z}_n^*$ can be translated to an unknown $\{x_{klj}\}_{k \in [s], l \in [s_k]}$ over $\prod_{k=1}^{s}(\prod_{l=1}^{s_k} \mathbb{Z}_{p_k^{\beta_{kl}}})$, where each x_{klj} is an unknown over $\mathbb{Z}_{p_k^{\beta_{kl}}}$ and analogously for the d_i's. Now each equation $\prod_{j=1}^{t} x_j^{a_{ij}} \equiv d_i \mod n$ can be translated to the a set of independent equations, namely $\{\sum_{j=1}^{t} a_{ij} x_{klj} \equiv d_{kli} \mod p_k^{\beta_l}\}_{k \in [s], l \in [s_k]}$

Let $E_{kl} = \{\sum_{j=1}^{t} a_{ij}x_{klj} \equiv d_{kli} \mod p_k^{\beta_l}\}_{i\in[t]}$. The bijection of the groups and the independence of each E_{kl} implies that

$$|\mathcal{S}(E)| = \prod_{k\in[s],l\in[s_k]} |\mathcal{S}(E_{kl})|. \tag{13}$$

Therefore, let us see how to bound the Solution Set of each E_{kl}.

The key observation is that Cramer's Rule still holds over linear equations modulo some integer. Namely, in our case, if $A = \{a_{ij}\}_{i,j\in[t]}$ and $A_j = \{a_{jfg}\}_{f,g\in[t]}$, where $a_{jfg} = \begin{cases} a_{fg} & \text{if } g \neq j \\ d_j & \text{if } g = j \end{cases}$ and $(\bar{x}_{kl1}, \ldots, \bar{x}_{klt}) \in \mathcal{S}(E_{kl})$, then

$$\bar{x}_j \cdot \det A \equiv \det A_j \mod p_k^{\beta_{kl}}$$

Therefore $|\mathcal{S}(E_{kl})| \subseteq |\mathcal{S}(\{\bar{x}_j \cdot \det A \equiv \det A_j \mod p_k^{\beta_{kl}}\}_{j\in[t]})|$. Since each equation $\bar{x}_j \cdot \det A \equiv \det A_j \mod p_k^{\beta_{kl}}$ has at most $\gcd(\det A, p_k^{\beta_{kl}})$ solutions, it holds that $|\mathcal{S}(E_{kl})| \leq (\gcd(\det A, p_k^{\beta_{kl}}))^t$. Hence, Equation (13) yields

$$|\mathcal{S}(E)| \leq \left(\prod_{k\in[s],l\in[s_k]} \gcd(\det A, p_k^{\beta_{kl}})\right)^t \tag{14}$$

Now, if there exists a $k^* \in [s]$ such that $\gcd(\det A, p_{k^*}) = 1$, then

$$|\mathcal{S}(E)| \leq \left(\prod_{k\in[s]\setminus\{k^*\},l\in[s_k]} \gcd(\det A, p_k^{\beta_{kl}})\right)^t \leq \left(\frac{\phi(n)}{p_{k^*}^{\beta_{k^*l}}}\right)^t \leq \left(\frac{\phi(n)}{p_{k^*}}\right)^t \tag{15}$$

Furthermore, if $(\det A, c)$ is not a factoring advice for n and $\det A \neq 0$, then there exists a prime $r > c$ such that r is a factor of $\phi(n)$ but not a factor of $\det A$. Therefore, we can view r as p_{k^*} and Equation (15) implies that indeed $|\mathcal{S}(E)| \leq (\frac{\phi(n)}{r})^t$.

5.9.2 Proof of Claim 5.4

In the case of system E^π, we observe that $Q_{C,G_\pi,q}$ is the corresponding matrix A from Lemma 5.3. Moreover, it holds $\det(Q_{C,G_\pi,q}) \neq 0$. This holds since $Q_{C,G_\pi,q} \equiv M_{\mathcal{I}}^{(H;C)^{G^\pi},H} \mod e$ (see Definition 4.15) and by assumption $\det(M_{\mathcal{I}}^{(H;C)^{G^\pi},H}) \mod e \neq 0$ (otherwise Forger would return $\perp$), we also have that $\det(Q_{C,G_\pi,q}) \mod e \neq 0$, yielding that $\det(Q_{C,G_\pi,q}) \neq 0$.

Furthermore, since $(\det(Q_{C,G_\pi,q}), |C|^4)$ is not a factoring advice for n, we conclude that $|\mathcal{S}(\mathrm{E}^\pi)| \leq (\frac{\phi(n)}{|C|^4})^t$.

5.9.3 Proof of Lemma 5.16

By Shannon-Fano encoding we have that if X is a random variable taking values in a finite set $\mathcal{U}$, then there exists a prefix-free code which describes each $u \in \mathcal{U}$ spending at most $\log \Pr[X = u] + 1$ bits. Now if we let $\mathcal{U} = \{\mathcal{Y}_k\}_{k=1}^m$, since these are disjoint subsets of $\mathcal{Y}$, we can set $\Pr[X = \mathcal{Y}_k] = \frac{|\mathcal{Y}_k|}{|\mathcal{Y}|}$ and the lemma follows.

5.9.4 Proving Claim 5.10

Before proving this claim let us see some useful definitions and facts.

Definition 5.17. *Let $E = \{e_1, \ldots, e_\ell\}$ be a set where e_j is an equation over $\mathbb{Z}_n^*$ of the form*

$$\prod_{i=1}^{t} x_i^{a_{ij}} \equiv c_j \mod n,$$

with $x_1, \ldots, x_t$ the unknowns. We denote by $\mathrm{Hom}(E)$ the corresponding homogeneous set of equations, i.e.

$$\prod_{i=1}^{t} x_i^{a_{ij}} \equiv 1 \mod n$$

The following lemma shows the connection between a set of equations E and its homogeneous set $\mathrm{Hom}(E)$:

Lemma 5.18. *Let E, be a set of equations as in Definition 5.17, then $|\mathcal{S}(E)| = |\mathcal{S}(\mathrm{Hom}(E))|$*

Proof. We will prove the following: Let $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_t) \in \mathcal{S}(E)$, then $\mathcal{S}(E) = \{\bar{y} \circ x_0 \colon x_0 \in \mathcal{S}(\mathrm{Hom}(E))\}$, where '$\circ$' denotes the group operation of $\mathbb{Z}_n^{*(t)}$. Let $\bar{z} = (\bar{z}_1, \ldots, \bar{z}_t) \in \mathcal{S}(E)$, then by assumption we know that for every equation $\prod_{i=1}^{t} x_i^{a_{ij}} \equiv c_j \mod n$ of E we have that $\prod_{i=1}^{t} \bar{y}_i^{a_{ij}} \equiv c_j \mod n$ and $\prod_{i=1}^{t} \bar{z}_i^{a_{ij}} \equiv c_j \mod n$. Consequently, we have that $\prod_{i=1}^{t} (\bar{y}_i^{-1} \cdot \bar{z}_i)^{a_{ij}} \equiv 1 \mod n$, which implies that $\bar{y}^{-1} \circ \bar{z} \in \mathcal{S}(\mathrm{Hom}(E))$ (where $\bar{y}^{-1}$ is the group inverse of $\bar{y}$ in $\mathbb{Z}_n^{*(t)}$). If we set $x_0 = \bar{y}^{-1} \circ \bar{z}$ we have that $\bar{z} = \bar{y} \circ s_H$. Moreover, if x_0, x_0' are two distinct elements of $\mathcal{S}(\mathrm{Hom}(E))$, then $\bar{y} \circ x_0 \neq \bar{y} \circ x_0'$, otherwise $x_0 = x_0'$. We conclude that $|\mathcal{S}(E)| = |\mathcal{S}(\mathrm{Hom}(E))|$. $\square$

The next lemma shows an interesting property of $\mathrm{Hom}(E)$, that we will use to measure $\mathcal{S}(\mathrm{Hom}(E))$ and thus $\mathcal{S}(E)$ as well.

Lemma 5.19. *If E is a set of equations over $\mathbb{Z}_n^*$ as in Definition 5.17, then $\mathcal{S}(\mathrm{Hom}(E))$ is a subgroup of $\mathbb{Z}_n^{*(t)}$.*

Proof. The unit vector of $\mathbb{Z}_n^{*(t)}$ trivially belongs to $\mathcal{S}(\mathrm{Hom}(E))$. Let $\bar{y} = (\bar{y_1}, \ldots, \bar{y_t})$, $\bar{z} = (\bar{z_1}, \ldots, \bar{z_t}) \in \mathcal{S}(\mathrm{Hom}(E))$. By definition of $\mathcal{S}(\mathrm{Hom}(E))$ we have that for every equation $\prod_{i=1}^{t} x_i^{a_{ij}} \equiv 1 \mod n$ of $\mathcal{S}(\mathrm{Hom}(E))$, $\prod_{i=1}^{t} \bar{y_i}^{a_{ij}} \equiv 1 \mod n$ and $\prod_{i=1}^{t} \bar{z_i}^{a_{ij}} \equiv 1 \mod n$. If we multiply these equations we have that $\prod_{i=1}^{t} (\bar{y_i} \cdot \bar{z_i})^{a_{ij}} \equiv 1 \mod n$, which shows that $\bar{y} \cdot \bar{z} \in \mathcal{S}(\mathrm{Hom}(E))$. Moreover, $\prod_{i=1}^{t} (\bar{y_i}^{-1})^{a_{ij}} \equiv 1^{-1} \equiv 1 \mod n$, which shows that $\bar{y}^{-1}$ also belongs to $\mathcal{S}(\mathrm{Hom}(E))$. The claim follows. $\square$

Therefore Lagrange Theorem yields the following:

Corollary 5.20. *Let E be a set of equations as in Definition 5.17 and $E' \subseteq E$, then*

$$\frac{|\mathcal{S}(\mathrm{Hom}(E \setminus E'))|}{|\mathcal{S}(\mathrm{Hom}(E))|} \in \mathbb{N}.$$

Proof. It follows from the fact that $\mathcal{S}(\mathrm{Hom}(E))$ is a subgroup of $\mathcal{S}(\mathrm{Hom}(E \setminus E'))$. $\square$

The next lemma shows that if E has few equations which reduce $|\mathcal{S}(E)|$ by a large factor, then most of these reduction is done by "few" equations.

Lemma 5.21. *Let E be a set of equations as in Definition 5.17, E_0 a subset of E such that $|\mathcal{S}(E_0)| = (\frac{\phi(n)}{\alpha \cdot r})^t$ for some prime $r|\phi(n)$ and some integer α and let $E|_i = \{e_1, \ldots, e_i\}$, then there exist t indices $i_1, \ldots, i_t$ such that*

$$\prod_{j=1}^{t} \frac{|\mathcal{S}(E|_{i_j-1})|}{|\mathcal{S}(E|_{i_j})|} \geq r^t$$

Proof. Let $\alpha_i = \frac{|\mathcal{S}(E|_{i-1})|}{|\mathcal{S}(E|_i)|}$ for $i \in [\ell]$ and $\alpha_0 = \phi(n)^t$. By Lemma 5.18 we have that $\alpha_i = \frac{|\mathcal{S}(\mathrm{Hom}(E|_{i-1}))|}{|\mathcal{S}(\mathrm{Hom}(E|_i))|}$ and by Corollary 5.20 it has to be an integer. As $\mathrm{E}_0 \subseteq E$ we know that $|\mathcal{S}(E)| = \frac{\phi(n)^t}{\alpha' \cdot r^t}$ for some $\alpha' \geq \alpha$, which has to be an integer because $|\mathcal{S}(E)| = |\mathcal{S}(\mathrm{Hom}(E))|$ and $\mathcal{S}(\mathrm{Hom}(E))$ is a subgroup of $\mathbb{Z}_n^{*(t)}$. In other words $\prod_{i=1}^{\ell} \alpha_i = \alpha' \cdot r^t$. The primarily of r yields the existence of a set $\mathcal{J} \subseteq \{1, \ldots, \ell\}$ of size t such that $\prod_{j \in \mathcal{J}} \alpha_j = \beta r^t$, for some integer β. The indices we are looking for are these in $\mathcal{J}$. □

Now we want to see what happens to the solution set if E not only has equations, but inequalities as well. We split the inequalities into two categories.

Definition 5.22. *Let E be a set of equations as in Definition 5.17, s_c be an equation of the form $\prod_{i=1}^{t} x_i^{a'_i} \equiv c \mod n$ and $\tilde{s_c}$ the corresponding inequality, namely $\prod_{i=1}^{t} x_i^{a'_i} \not\equiv c \mod n$. Then $\tilde{s_c}$ is for E of*

- *r-Type I: if $\frac{|\mathcal{S}(E)|}{|\mathcal{S}(E \cup \{s_c\})|} < r$*
- *r-Type II: otherwise*

Remark 5.23. *Notice that given E the Type of an inequality $\tilde{s_c}$ of the form $\prod_{i=1}^{t} x_i^{a'_i} \not\equiv c \mod n$, only depends on the exponents a'_i's and not on the right hand side c. This holds because for every $c, c' \in \mathbb{Z}_n^*$ $|\mathcal{S}(E \cup \{s_c\})| = |\mathcal{S}(\mathrm{Hom}(E \cup \{s_c\}))| = |\mathcal{S}(\mathrm{Hom}(E \cup \{s_{c'}\}))| = |\mathcal{S}(E \cup \{s_{c'}\})|$.*

The following lemma is useful.

Lemma 5.24. *Let E be a set of equations in Definition 5.17, s_1, s_2 two equations over $\mathbb{Z}_n^*$ over the same unknowns, r a prime and a, b, c the largest integers such that $r^a | \, |\mathcal{S}(E)|$, $r^b | \, |\mathcal{S}(E \cup \{s_1\}|$ and $r^c | \, |\mathcal{S}(E \cup \{s_2\}|$. If $a = b$, then $r^c | \, |\mathcal{S}(E \cup \{s_1\} \cup \{s_2\})|$, with c being the largest such integer.*

Proof. Again we are going to use that $|\mathcal{S}(E)| = |\mathcal{S}(\mathrm{Hom}(E))|$. As $\mathcal{S}(\mathrm{Hom}(E))$ a group of size say m, by the Fundamental Theorem of Abelian Groups we know (as in the proof of Lemma 5.3) that if $m = r^a \prod_{i=1}^{s} p_i^{\alpha_i}$ is the factorization of m, then $\mathcal{S}(\mathrm{Hom}(E)) \cong V \times W$, where $V \cong \prod_{j=1}^{v} \mathbb{Z}_{r^{a_j}}$, with $\sum_{j=1}^{v} a_j = a$ and $W \cong \prod_{i=1}^{s} (\prod_{k=1}^{s_i} \mathbb{Z}_{p_i^{\alpha_{ik}}})$, with $\sum_{k=1}^{s_i} \alpha_{ik} = \alpha_i$. Thus, we have that $|V| = r^a$ and $|W| = \prod_{i=1}^{s} p_i^{\alpha_i}$. Analogously, we have

- $\mathcal{S}(\mathrm{Hom}(E \cup \{s_1\})) \cong V' \times W'$
- $\mathcal{S}(\mathrm{Hom}(E \cup \{s_2\})) \cong V'' \times W''$
- $\mathcal{S}(\mathrm{Hom}(E \cup \{s_1, s_2\})) \cong V''' \times W'''$

with $|V'| = r^b$ and $|V''| = r^c$ and $\gcd(r, |W'| \cdot |W''| \cdot |W'''|) = 1$.

Since $\mathcal{S}(\mathrm{Hom}(E \cup \{s_1, s_2\})) = \mathcal{S}(\mathrm{Hom}(E \cup \{s_1\})) \cap \mathcal{S}(\mathrm{Hom}(E \cup \{s_2\}))$, it follows that $V''' \equiv V' \cap V''$ and $W''' \equiv W' \cap W''$. However, if $a = b$, then $V \equiv V'$, which implies that $V''' = V \cap V''$. Since $\mathcal{S}(\mathrm{Hom}(E \cup \{s_2\})) \subseteq \mathcal{S}(\mathrm{Hom}(E))$, we have that $V'' \subseteq V$ and therefore $V''' \equiv V''$. Thus, $|V'''| = r^c$ and we conclude that $r^c|\ |\mathcal{S}(\mathrm{Hom}(E \cup \{s_1, s_2\}))|$. To prove that c is the largest such integer, we have to show that $|W' \cap W''|$ is not a multiple of r. But r is not a multiple of $|W'|$ (nor of $|W''|$) and $W' \cap W''$ is a subgroup of W' (and of W''), thus by Lagrange's Theorem r is not a multiple of $|W' \cap W''|$ either. The lemma follows from the fact that $|\mathcal{S}(E \cup \{s_1, s_2\})| = |\mathcal{S}(\mathrm{Hom}(E \cup \{s_1, s_2\}))|$. □

The next lemma shows that inequalities do not decrease the solution set by more than a factor of 2.

Lemma 5.25. *Let E be as in Definition 5.17, $\tilde{E}$ a set of inequalities and e an equation over the same unknowns with E, r a prime factor of $|\mathcal{S}(E)|$, such that $r^\tau|\ \frac{|\mathcal{S}(E)|}{|\mathcal{S}(E\cup\{e\})|}$, where τ the largest such integer and $r \geq 2 \cdot |\tilde{E}|$. Then*

1. *If $\tilde{E}$ only contains inequalities of r-Type I, then $r^\tau|\ \frac{|\mathcal{S}(E\cup\tilde{E})|}{|\mathcal{S}(E\cup\tilde{E}\cup\{e\})|}$ and*
2. *If $\tilde{E}$ also contains inequalities of r-Type II, then $\frac{r^\tau}{2} \leq \frac{|\mathcal{S}(E\cup\tilde{E})|}{|\mathcal{S}(E\cup\tilde{E}\cup\{e\})|}$.*

Proof.

1. Let us prove it for the case $|\tilde{E}| = 1$. For every tuple $s = (b_1, \dots, b_t) \in \mathbb{Z}_n^{*(t)}$ let s_c denote an equality of the form $\prod_{i=1}^t x_i^{b_i} \equiv c \mod n$ and let $\tilde{s_c}$ be the corresponding inequality, (i.e. $\prod_{i=1}^t x_i^{b_i} \not\equiv c \mod n$). Let C_E^s be the set with all elements $c \in \mathbb{Z}_n^*$ such that s_c is consistent with E, namely there exists a tuple $(\bar{x}_1, \dots, \bar{x}_t) \in \mathcal{S}(E)$ such that $\prod_{i=1}^t \bar{x}_i^{b_i} \equiv c \mod n$. By the definition of C_E^s we have:

   (a) $\mathcal{S}(E) = \bigcup_{c \in C_E^s} \mathcal{S}(E \cup \{s_c\})$

   (b) For every distinct $c, c' \in C_E^s$, $\mathcal{S}(E \cup \{s_c\}) \cap \mathcal{S}(E \cup \{s_{c'}\}) = \emptyset$ and

   (c) $|\mathcal{S}(E \cup \{s_c\})| = |\mathcal{S}(E \cup \{s_{c'}\})| = |\mathcal{S}(\mathrm{Hom}(E \cup \{s_1\}))|$.

   From property (a) we can see that for every $c \in C_E^s$ we have that $\mathcal{S}(E \cup \{\tilde{s}_c\}) = \bigcup_{c' \in C_E^s \setminus \{c\}} \mathcal{S}(E \cup \{s_{c'}\})$. Therefore, by Lemma 5.18 and properties (b),(c) it holds that if $\tilde{E} = \{\tilde{s}_c\}$ all we have to show is that $r^\tau|\ \frac{|\mathcal{S}(E\cup\{s_c\})|}{|\mathcal{S}(E\cup\{s_c\}\cup\{e\})|}$, because $\frac{|\mathcal{S}(E\cup\{s_c\})|}{|\mathcal{S}(E\cup\{s_c\}\cup\{e\})|} = \frac{\sum_{c' \in C_E^s \setminus \{c\}} |\mathcal{S}(E\cup\{s_{c'}\})|}{\sum_{c' \in C_E^s \setminus \{c\}} |\mathcal{S}(E\cup\{s_{c'}\}\cup\{e\})|} = \frac{|\bigcup_{c' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c'}\})|}{|\bigcup_{c' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c'}\}\cup\{e\})|} = \frac{|\mathcal{S}(E\cup\{\tilde{s_c}\})|}{|\mathcal{S}(E\cup\{\tilde{s_c}\}\cup\{e\})|}$. As $\tilde{s}_c$ is of r-Type I and r prime, if r^τ divides $|\mathcal{S}(E)|$ then r^τ also divides $|\mathcal{S}(E \cup \{s_c\})|$ and τ is the largest such integer. By Lemma 5.24, if τ' is the largest integer such that $r^{\tau'}|\ |\mathcal{S}(E \cup \{e\})|$, then τ' is also the largest integer such that $r^{\tau'}|\ |\mathcal{S}(E \cup \{s_c\} \cup \{e\})|$. The claim follows.

   For the case of $|\tilde{E}| = 2$ with $\tilde{E} = \{\tilde{s}_c, \tilde{s}'_{c'}\}$, we just observe that by Lemma 5.24, $\tilde{s}'_{c'}$ is also an r-Type I inequality for $E \cup \{s_c\}$. Therefore, $r^t|\ \frac{|\mathcal{S}(E\cup\{s_c\}\cup\{\tilde{s}'_{c'}\})|}{|\mathcal{S}(E\cup\{s_c\}\cup\{\tilde{s}'_{c'}\}\cup\{e\})|} = \frac{|\sum_{c'' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c''}\}\cup\{\tilde{s}'_{c'}\})|}{|\sum_{c'' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c''}\}\cup\{\tilde{s}'_{c'}\}\cup\{e\})|} = \frac{|\bigcup_{c'' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c''}\}\cup\{\tilde{s}'_{c'}\})|}{|\bigcup_{c'' \in C_E^s \setminus \{c\}} \mathcal{S}(E\cup\{s_{c''}\}\cup\{\tilde{s}'_{c'}\}\cup\{e\})|} = \frac{|\mathcal{S}(E\cup\{\tilde{s}_c,\tilde{s}'_{c'}\})|}{|\mathcal{S}(E\cup\{\tilde{s}_c,\tilde{s}'_{c'}\}\cup\{e\})|} = \frac{|\mathcal{S}(E\cup\tilde{E})|}{|\mathcal{S}(E\cup\tilde{E}\cup\{e\})|}$. For the general case we continue arguing in the same way.

2. Let $\tilde{E}$ only contain r-Type II inequalities. it follows that $|\mathcal{S}(E \cup \tilde{E}))| \geq (1 - \frac{|\tilde{E}|}{r}) \cdot |\mathcal{S}(E)| \geq \frac{1}{2}|\mathcal{S}(E)|$. By the assumption of $\tilde{E}$ we have that $|\mathcal{S}(E)| \geq r^{\tau} \cdot |\mathcal{S}(E \cup \{e\})| \geq r^{\tau} \cdot |\mathcal{S}(E \cup \tilde{E} \cup \{e\})|$. Combining them we get $\frac{r^{\tau}}{2} \leq \frac{|\mathcal{S}(E \cup \tilde{E})|}{|\mathcal{S}(E \cup \tilde{E} \cup \{e\})|}$.

   For the general case, we see that $\mathcal{S}(E \cup \tilde{E})$ can be written as $\mathcal{S}(E \cup \tilde{\mathrm{E}_I} \cup \tilde{\mathrm{E}_{II}})$, where $\tilde{\mathrm{E}_I}$ only contains inequalities of r-Type I and $\tilde{\mathrm{E}_{II}}$ only inequalities of r-Type II. The above yields that for each inequality $\tilde{s}_c$, $\mathcal{S}(E \cup \{\tilde{s}_c\})$ can be written as a union of equally sized disjoint sets, i.e. $\bigcup_{c' \in C^s_E \setminus \{c\}} \mathcal{S}(E \cup \{s_{c'}\})$. Therefore $\mathcal{S}(E \cup \tilde{E})$ can be written as $\bigcup_{i \in \mathcal{I}} \mathcal{S}(E \cup E_i)$, for some sets $\{E_i\}_{i \in \mathcal{I}}$ that contain only equations and the sets $\mathcal{S}(E \cup E_i)$ are equally sized and disjoint. Now we have

   $\frac{r^t}{2} \leq \frac{|\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}})|}{|\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}} \cup \{e\})|}$, but $\frac{|\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}})|}{|\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}} \cup \{e\})|} = \frac{\sum_{i \in \mathcal{I}} |\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}})|}{\sum_{i \in \mathcal{I}} |\mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}} \cup \{e\})|} = \frac{|\bigcup_{i \in \mathcal{I}} \mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}})|}{|\bigcup_{i \in \mathcal{I}} \mathcal{S}(E \cup \mathrm{E}_i \cup \tilde{\mathrm{E}_{II}} \cup \{e\})|} = \frac{|\mathcal{S}(E \cup \tilde{\mathrm{E}_I} \cup \tilde{\mathrm{E}_{II}})|}{|\mathcal{S}(E \cup \tilde{\mathrm{E}_I} \cup \tilde{\mathrm{E}_{II}} \cup \{e\})|} = \frac{|\mathcal{S}(E \cup \tilde{E})|}{|\mathcal{S}(E \cup \tilde{E} \cup \{e\})|}$.

   □

Now we are ready to prove Claim 5.10.

Proof of Claim 5.10. Let Q be number of queries of $(H; C)$. Since E^{π} is a subset of E^{π}_Q and since by Claim 5.4 $|\mathcal{S}(\mathrm{E}^{\pi})| \leq (\frac{\phi(n)}{r})^t$ for some prime $r|\phi(n)$ and $r \geq |C|^4$, by Lemma 5.21 we have that there exist t colliding queries such that $\prod_{j=1}^{t} \frac{|\mathcal{S}(\mathrm{E}^{\pi}_{i_j-1} \setminus \tilde{\mathrm{E}}^{\pi}_{i_j-1})|}{|\mathcal{S}(\mathrm{E}^{\pi}_{i_j} \setminus \tilde{\mathrm{E}}^{\pi}_{i_j})|} \geq (|C|^4)^t$, where $\tilde{\mathrm{E}}^{\pi}_j$ is the set of all inequalities derived in the first j queries. As for each $\tilde{\mathrm{E}}^{\pi}_j$, we have that $|\tilde{\mathrm{E}}^{\pi}_j| \leq 4|C|^2$ (an upper bound on $|\tilde{\mathrm{E}}^{\pi}_j|$) we have that $|C|^4 \geq 2 \cdot |\tilde{\mathrm{E}}^{\pi}_j|$ and we can use Lemma 5.25 . The claim follows. □

6 A t-EU-CMA-secure RSA-FDH Signature Scheme

In this section we show the following under the RSA assumption (see below): for every polynomially-bounded integer function t there exists an efficient procedure KeyGen_t for which RSA-FDH scheme $\Sigma_t = (\mathsf{KeyGen}_t, \mathsf{Sign}, \mathsf{Verify})$ is t-EU-CMA-secure. Furthermore, the reduction treats $\mathbb{Z}^*_n$ and the potential adversary in a black-box way.

Definition 6.1 (RSA Assumption (in the Standard Model))**.** *There exist polynomial time procedures* RSAKey *and* $\mathsf{RSAChallenge}$ *such that*

1. RSAKey *on input* 1^k *outputs an integer* (n, p_1, p_2, e, d)*, such that* $n = p_1 p_2$*, with* $p_1, p_2 \in \mathrm{P}$*,* $\gcd(e, \phi(n)) = 1$ *and* $ed \equiv 1 \mod \phi(n)$*,*
2. $\mathsf{RSAChallenge}$ *on input* (n, e) *outputs* $y \in \mathbb{Z}_n$*, and*
3. *for any* PPT A *it holds that*

$$\mathsf{Pr}_{(n,p_1,p_2,e,d) \leftarrow \mathsf{RSAKey}(1^k), y \leftarrow \mathsf{RSAChallenge}(n,e)}[x^e \equiv y \mod n \colon x \leftarrow A(n, e, y)] = \mathrm{neg}(k)$$

To construct the hash function of our RSA-FDH scheme, we use families of *cover free subsets*.

Definition 6.2 (cover-free subsets)**.** *Let* $\mathcal{S}_1, \ldots, \mathcal{S}_n$ *be subsets of a universe* $\mathcal{U}$*. These subsets are called* t-cover free *if for any set of indices* $\{i_1, \ldots, i_t\}$ *and for any index* i_0 *such that* $i_0 \notin \{i_1, \ldots, i_t\}$*, it holds that* $\mathcal{S}_{i_0} \not\subseteq \bigcup_{i \in \{i_1, \ldots, i_t\}} \mathcal{S}_i$*.*

The existence of such cover free subsets was proven by D'yachkov et al. [13], where the fact that we can construct them in polynomial time was proven by Kumar et al. [23].

Theorem 6.3 ([13, 23])**.** *There exist a polynomial-time algorithm* CFGen *and a constant* $c > 0$ *such that the following holds: for any integers* n *and* t *and a set* $\mathcal{U}$ *of size* $\lceil c \cdot t^2 \log n \rceil$*, the family of subsets* $\mathcal{S}_1, \ldots, \mathcal{S}_n \subset \mathcal{U}$*, where* $\mathcal{S}_i = \mathsf{CFGen}(n, t, \mathcal{U}, i)$*, is* t*-cover free.*

Given a polynomial bounded integer function t, we assume without loss of generality that $t(k) = \text{poly}(k)$, and define the t-secure RSA-FDH signature scheme $\Sigma_t = (\mathsf{KeyGen}_t, \mathsf{Sign}, \mathsf{Verify})$ as follows (we only define the key generator KeyGen_t, since the signer and verifier Sign and Verify are generic):

Algorithm 6.4 (KeyGen_t)**.**

Input: 1^k.

Operation:

1. *Let* $(n, p_1, p_2, e, d) \leftarrow \mathsf{RSAKey}(1^k)$.
2. *Let* $u = \lceil ct(k)^2 \log n \rceil$*, where* c *is the constant from Theorem 6.3, and for* $i \in [u]$ *let* $a_i \leftarrow \mathsf{RSAChallenge}(n, e)$ *and set* $\mathcal{U} = \{a_i\}_{i \in [u]}$.
3. *Construct the function* $h_{\mathcal{U}} : \mathbb{Z}_n \mapsto \mathbb{Z}_n$*, defined as* $h_{\mathcal{U}}(m) = \prod_{i \in \mathcal{S}_m} a_i \bmod n$*, where* $\mathcal{S}_m := \mathsf{CFGen}(n, t, \mathcal{U}, m)$.
4. *Return* $pk = (n, e, h_{\mathcal{U}})$ *and* $sk = d$.

Theorem 6.5 (Theorem 1.1, restated)**.** *Under the* RSA *assumption* Σ_t *is* t-EU-CMA-*secure.*

Proof. Suppose that there exists an adversary F, that breaks $t(k)$-times security of Σ_t with some non negligible probability $\delta(k)$. We show that there exists an PPT A which given black-box access to F breaks the RSA assumption. More specifically, A will be such that

$$\mathsf{Pr}_{(n,p_1,p_2,e,d) \leftarrow \mathsf{RSAKey}(1^k), y \leftarrow \mathsf{RSAChallenge}(n,e)}[x^e \equiv y \mod n | x \leftarrow A^F(n, e, y)] \geq \Theta\left(\frac{\delta(k)}{t(k)^2 \log n}\right)$$

A is described as follows:

Algorithm 6.6. A

Input: $(1^k, n, e, y)$

Operation:

1. *Let* $u = \lceil ct(k)^2 \log n \rceil$*, where* c *is the constant from Theorem 6.3, and for* $i \in [u]$ *let* $b_i \leftarrow \mathsf{RSAChallenge}(n, e)$.
2. *For* $i = 1, \ldots, u$ *set* $a_i = b_i^e \mod n$.
3. *If* $a_i \equiv y \mod n$ *for some* $i \in [u]$*, then return* b_i.

4. *Choose* $\ell \leftarrow [u]$ *uniformly at random.*

5. *Set* $a_\ell :\ = y$ *and* $\mathcal{U} = \{a_i\}_{i \in [u]}$

6. *Query* F *on* $(n, e, h_\mathcal{U})$, *where* $h_\mathcal{U}$ *is as defined in* 6.4.

7. *For each signing query of* F *on message* m_i, *where* $i \in [t(k)]$ *do*

   (a) *If* $\ell \notin \mathcal{S}_{m_i}$ *respond to* F *with* $\sigma_i = \prod_{i \in \mathcal{S}_{m_i}} b_i \mod n$ *(where* $\mathcal{S}_{m_i}$ *is as defined in* 6.4*).*

   (b) *Else abort.*

8. *If* F *returns a valid forgery* σ *on message* m *and* $\ell \in \mathcal{S}_m$ *return* $\sigma \cdot \prod_{i \in \mathcal{S}_m \setminus \{\ell\}} b_i^{-1} \mod n$

If A stops at step 3 then it clearly succeeds. Below we assume that this is not the case.

To show that A^F succeeds with probability $\Theta(1/t(k)^2 \log n)$, consider the following experiment. Choose u random elements $a_1, \ldots, a_u$ of $\mathbb{Z}_n$ and a random index $\ell \in [u]$. Query F on a random input $(n, e, h_\mathcal{U})$ chosen according to $\mathsf{KeyGen}_t(1^k)$ and answer all signing requests. The success condition of this experiment is that F (1) succeeds in its forgery, (2) it requests the signatures of $m_1, \ldots, m_{t(k)}$ such that $d \notin \bigcup_{j \in \{m_1, \ldots, m_{t(k)}\}} \mathcal{S}_j$ and (3) outputs the signature of some m with $d \in \mathcal{S}_m$. If F succeeds we know that $m \notin \{m_1, \ldots, m_{t(k)}\}$ and by the definition of subsets there exists one ℓ' such that $\ell' \notin \bigcup_{j \in \{m_1, \ldots, m_{t(k)}\}} \mathcal{S}_j$ but $d' \in \mathcal{S}_m$. By assumption F succeeds in its forgery with probability $\delta(k)$ and with probability $1/u = \Theta(1/t(k)^2 \log n)$ we have that $\ell = \ell'$. Since these conditions are independent, the probability of success in the above experiment is $\Theta(\delta(k)/t(k)^2 \log n)$. If $\delta(k)$ is non-negligible then the latter probability is also non-negligible.

We argue that the probability of success of A is at least the probability of success of the above experiment. The input to F follows the same distribution as the output of $\mathsf{KeyGen}_t(1^k)$. Whenever, F requests a signature σ_{m_i} of a message m_i such that $\ell \notin \mathcal{S}_{m_i}$, A returns a valid signature of m_i (since $(\prod_{i \in \mathcal{S}_{m_i}} b_i)^e \equiv \prod_{i \in \mathcal{S}_{m_i}} b_i^e \equiv \prod_{i \in \mathcal{S}_{m_i}} a_i \mod n$). If F forges m with a valid signature σ and $\ell \in \mathcal{S}_m$, then we have that $(\sigma \cdot \prod_{i \in \mathcal{S}_m \setminus \{\ell\}} b_i^{-1})^e \equiv h_\mathcal{U}(m) \cdot \prod_{i \in \mathcal{S}_m \setminus \{\ell\}} a_i^{-1} \equiv \prod_{i \in \mathcal{S}_m} a_i \cdot \prod_{i \in \mathcal{S}_m \setminus \{\ell\}} a_i^{-1} \equiv a_\ell \equiv y \mod n$. □

Acknowledgments

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

References

[1] D. Aggarwal and U. M. Maurer. Breaking RSA generically is equivalent to factoring. In *Advances in Cryptology – EUROCRYPT 2009*, 2009.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology – EUROCRYPT '94*, pages 92–111, 1994.

[4] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Advances in Cryptology – EUROCRYPT 2004*, 2004.

[5] A. Boldyreva and M. Fischlin. Analysis of random oracle instantiation scenarios for oaep and other practical schemes. In *Advances in Cryptology – CRYPTO 2005*, pages 412–429, 2005.

[6] J. Brown, J. M. G. Nieto, and C. Boyd. Efficient cca-secure public-key encryption schemes from rsa-related assumptions. In *INDOCRYPT 2006*, pages 176–190, 2006.

[7] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *JACM: Journal of the ACM*, 51, 2004.

[8] R. Canetti, O. Goldreich, and S. Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *Theory of Cryptography, First Theory of Cryptography Conference (TCC)*, 2004.

[9] J. Coron. On the exact security of full domain hash. In *Advances in Cryptology – CRYPTO 2000*, 2000.

[10] R. Cramer and V. Shoup. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[11] J. D. Dixon. Asymptotically fast factorization of integers. *Mathematics of Computation*, 36: 255–260, 1981.

[12] Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of the full domain hash. In *Advances in Cryptology – CRYPTO 2005*, pages 449–466, 2005.

[13] A. G. D'yachkov, P. A. Vilenkin, D. C. Torney, and A. J. Macula. Families of finite sets in which no intersection of sets is covered by the union of s others. *J. Comb. Theory, Ser. A*, 99 (2):195–218, 2002.

[14] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 305–313, 2000.

[15] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, pages 123–139, 1999.

[16] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal on Computing*, 35(1):217–246, 2005.

[17] S. Goldwasser and Y. Tauman-Kalai. On the (in)security of the fiat-shamir paradigm. In *Proceedings of the 44th Annual Symposium on Foundations of Computer Science (FOCS)*, 2003.

[18] I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In *Theory of Cryptography, Sixth Theory of Cryptography Conference (TCC)*, 2009.

[19] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. 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)*. IEEE Computer Society, 2007.

[20] D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In *Advances in Cryptology – ASIACRYPT 2011*, 2011.

[21] S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *Advances in Cryptology – CRYPTO 2009*, pages 654–670, 2009.

[22] E. Kiltz and K. Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In *Advances in Cryptology – EUROCRYPT 2009*, pages 389–406, 2009.

[23] R. Kumar, S. Rajagopalan, and A. Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology – CRYPTO '99*, 1999.

[24] U. M. Maurer. Abstract models of computation in cryptography. In *IMA Int. Conf.*, pages 1–12, 2005.

[25] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

[26] V. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *MATHNASUSSR: Mathematical Notes of the Academy of Sciences of the USSR*, 55, 1994.

[27] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO 2002*, 2002.

[28] P. Paillier. Impossibility proofs for RSA signatures in the standard model. In *CT-RSA*, Lecture Notes in Computer Science, pages 31–48, 2007.

[29] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Advances in Cryptology – ASIACRYPT 2005*, pages 1–20, 2005.

[30] P. Paillier and J. L. Villar. Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In *Advances in Cryptology – ASIACRYPT 2006*, pages 252–266, 2006.

[31] K. Pietrzak. Compression from collisions, or why CRHF combiners have a long output. In *Advances in Cryptology – CRYPTO 2009*, 2008.

[32] R. L. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[33] *PKCS #1: RSA Encryption Standard*. RSA Laboratories, Redwood City, California, Nov. 1993.

[34] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: 379–423 & 623–656, 1948.

[35] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, 1997.

[36] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[37] H. Wee. One-way permutations, interactive hashing and statistically hiding commitments. In *Theory of Cryptography, Fourth Theory of Cryptography Conference (TCC)*, pages 419–433, 2007.

[38] Y.Dodis and L.Reyzin. On the power of claw-free permutations. In *International Conference on Security in Communication Networks, SCN, LNCS*, volume 3, 2002.