J Cryptol (2018) 31:276–306
https://doi.org/10.1007/s00145-017-9257-9

Journal of
CRYPTOLOGY

Optimal Security Proofs for Full Domain Hash, Revisited

Saqib A. Kakvi
Paderborn University, Paderborn, Germany
saqib.kakvi@upb.de

Eike Kiltz
Ruhr-University Bochum, Bochum, Germany
eike.kiltz@rub.de

Communicated by Rogaway.

Received 24 June 2013
Online publication 24 April 2017

Abstract. RSA Full Domain Hash (RSA-FDH) is a digital signature scheme, secure against chosen message attacks in the random oracle model. The best known security reduction from the RSA assumption is non-tight, i.e., it loses a factor of q_s, where q_s is the number of signature queries made by the adversary. It was furthermore proven by Coron (Advances in cryptology—EUROCRYPT 2002, Lecture notes in computer science, vol 2332. Springer, Berlin, pp 272–287, 2002) that a security loss of q_s is optimal and cannot possibly be improved. In this work, we uncover a subtle flaw in Coron's impossibility result. Concretely, we show that it only holds if the underlying trapdoor permutation is *certified*. Since it is well known that the RSA trapdoor permutation is (for all practical parameters) not certified, this renders Coron's impossibility result moot for RSA-FDH. Motivated by this, we revisit the question whether there is a tight security proof for RSA-FDH. Concretely, we give a new tight security reduction from a stronger assumption, the Phi-Hiding assumption introduced by Cachin et al. (Advances in Cryptology—EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 402–414, 1999). This justifies the choice of smaller parameters in RSA-FDH, as it is commonly used in practice. All of our results (positive and negative) extend to the probabilistic signature scheme PSS (with message recovery).

Keywords. Digital signatures, Full domain hash, Lossiness, Security reduction.

1. Introduction

Among all digital signatures schemes based on the RSA problem, arguably among the most important ones is RSA Full Domain Hash (RSA-FDH) by Bellare and Rogaway [4]. It is extensively used in a wide variety of applications and serves as the basis of several existing standards such as PKCS #1 [29]. It has been demonstrated by means of a security reduction that, in the random oracle model [3], breaking the security of RSA-FDH (in the sense of existential unforgeability against chosen message attacks) is asymptotically at least as hard as inverting the RSA function.

The seminal work by Bellare and Rogaway introduced the concept of concrete security [4] and highlights the importance of considering the tightness of a security reduction. A security reduction is *tight* if an adversary breaking the scheme yields another adversary breaking the underlying hardness assumption with roughly the same success probability and running time. The current state of RSA-FDH is as follows. Coron's reduction [12] (which improves on earlier results by Bellare and Rogaway [4]) bounds the probability ε of breaking RSA-FDH in time t by $\varepsilon' \cdot q_s$, where ε' is the probability of inverting RSA in time $t' \approx t$ and q_s is the number of signature queries by the forger. In other words, the security reduction for RSA-FDH is loose (it loses a factor of q_s), which can have great negative impact on the practical parameter choices of the scheme. As a numerical example, for 80 bits of security and assuming that an adversary can make up to $q_s = 2^{30}$ signature queries [4], one should use a large enough RSA modulus N such that inverting the RSA function cannot be done in fewer than $2^{110} = 2^{30} \cdot 2^{80}$ operations. Concretely, using the recommended key sizes from [31], this leads to a modulus N of about 2432 bits, compared to 1248 bits if RSA-FDH had a tight reduction. We further refer to [23] for a recent discussion on the practical impact of non-tight security reductions in cryptography.

It is an interesting question of great practical impact whether or not there is a tight security reduction for general FDH signatures (based on any trapdoor permutation TDP) and, in particular, for RSA-FDH. Unfortunately, this question was already answered to the negative exactly 10 years ago by Coron [13,14] who showed that the above non-tight security reduction is essentially *optimal*. That is, every security reduction from inverting the TDP (i.e., RSA in the case of RSA-FDH) to breaking FDH signatures will inevitably lose a q_s factor. Consequently, for RSA-FDH a large RSA modulus seems unavoidable to obtain a meaningful security proof.

1.1. *An Overview of Our Results*

Revisiting Coron's Impossibility Result. We uncover a gap in Coron's result about the impossibility of a tight security reduction for FDH signatures [14]. As acknowledged by the author of [14], his impossibility result only holds if the underlying trapdoor permutation (i.e., RSA in the case of RSA-FDH) is a *certified trapdoor permutation*. A trapdoor permutation is certified [6,25] if one can publicly verify that it actually defines a permutation. However, in the case of the RSA trapdoor permutation, it is only known to be certified if the exponent e is a prime number larger than $N^{1/4}$ [21], and therefore, the impossibility result does not apply to all instances of RSA-FDH. In particular, it is common practice to use either $e = 3$ or $e = 2^{16} + 1$, which are less then $N^{1/4}$.

A Tight Security Reduction for FDH Signatures. In light of the above, we revisit the question whether there exists a tight security reduction for FDH signatures. Unfortunately, we are not able to give such a tight security reduction from the assumption that the TDP is one-way, but from a stronger (yet still non-interactive) assumption, namely that the TDP is lossy (in the sense of Peikert and Waters [28]). Our main result (Theorem 3) shows that there is a *tight* security reduction from the lossiness of the TDP to breaking security of FDH, in the random oracle model. Our results also extend to the probabilistic signature scheme (PSS) (with message recovery) [4]. We obtain a tight reduction for TDP-PSS with arbitrary (possibly zero) size random seed from the assumption that the TDP is lossy.

Applications to RSA-FDH and RSA-PSS. Recently, Kiltz et al. [22] showed that the RSA trapdoor permutation is lossy under the Φ-Hiding Assumption. The Φ-Hiding Assumption was introduced by Cachin et al. [10] and it states that, roughly, (N, e) with $\gcd(\varphi(N), e) = 1$ and $e < N^{1/4}$ is computationally indistinguishable from (N', e') with $e' \mid \varphi(N')$. (Here, $\varphi(N)$ is Euler's totient function.) This give a tight security reduction for RSA-FDH from the Φ-Hiding Assumption. We remark that the Φ-Hiding Assumption (or, more generally, the assumption that RSA is lossy) is a stronger assumption than the assumption that RSA is one-way. However, it dates back to 1999 [10] and has ever since been used in a number of cryptographic applications (e.g., [9,15,17,19,22,26]). It has been cryptalanyzed (e.g., [9,10,30]), and for the parameters of interest there is no known algorithm that breaks it without first factoring, the modulus $N = pq$. The common interpretation is that the Φ-Hiding Assumption can in practice be viewed as *as hard as factoring* and hence gives a theoretical justification as to why RSA-FDH with a small modulus N is secure in practice.

We also obtain a tight reduction for RSA-PSS (with message recovery) from the Φ-Hiding Assumption for arbitrary (possibly zero) size random seed. Assuming again that the Φ-Hiding Assumption is as hard as factoring, we get a signature scheme (with message recovery) with only 160 bits of overhead for 80 bits security.

1.2. *Full Domain Hash and Coron's Impossibility Result*

Recall that a FDH signature on a message m is $\sigma = f^{-1}(H(m))$, where f is the public description of the TDP and H is a hash function modeled as a random oracle.

A reduction $\mathcal{R}$ that reduces inverting the TDP to breaking FDH inputs a challenge instance $(f, y = f(x))$ of the TDP and generates a public key for FDH that is passed to a forger $\mathcal{F}$ attacking FDH signatures. Next, $\mathcal{F}$ makes a number of signature queries (which are answered by $\mathcal{R}$) and finally outputs a forgery. Finally, $\mathcal{R}$ uses the gathered information to invert the TDP, i.e., to compute $x = f^{-1}(y)$. Reduction $\mathcal{R}$ is *tight* if the success probability of $\mathcal{R}$ is roughly the same as the one of $\mathcal{F}$.

Coron's impossibility result shows that any reduction $\mathcal{R}$ from inverting the TDP f to breaking FDH which is tight (then, does not lose more than a factor q_s) can be turned into an efficient inverting algorithm $\mathcal{I}$ for the TDP f (that works without forger $\mathcal{F}$). In a nutshell, the argument is as follows. Given an instance of the TDP, the inverter $\mathcal{I}$ runs reduction $\mathcal{R}$ providing it with a simulated forger $\mathcal{F}$ by making a number of hash queries and then signature queries to $\mathcal{R}$. Next, $\mathcal{I}$ rewinds reduction $\mathcal{R}$ to an earlier state (after the hash queries) and uses one of the signed messages/signature pairs [say (m^*, σ^*)] obtained before the rewind as its forgery. To $\mathcal{R}$, this counts as a valid forgery since after the rewind, $\mathcal{I}$ did not make a signing query on m^*. The central argument is as follows: Consider a real forger that is provided with the same view as the simulated forger who outputs a forgery σ' on the same message m^*. FDH has *unique signatures*[1], and hence,

[1] A signature scheme has unique signatures if for each message there exists exactly one signature that verifies w.r.t. a given (honestly generated) public key.

we can argue that σ^* (provided by $\mathcal{R}$ before the rewind) equals σ' (provided by a real forger). Hence, $\mathcal{R}$ is convinced and interacts with a real forger and outputs a solution to the TDP instance.

Consequently, from $\mathcal{R}$, we were able to construct an algorithm $\mathcal{I}$ that inverts the TDP without using any forger. It is shown by a combinatorial argument that the success probability of $\mathcal{I}$ is nonnegative as long as the reduction $\mathcal{R}$ does not loose more than a factor of q_s, the number of signature queries.

THE GAP IN THE PROOF. During the proof of [13, Th. 5] it is silently assumed that the public key pk generated by reduction $\mathcal{R}$ is a *real public key*, honestly generated by the key-generation algorithm of FDH, i.e., it contains f which describes a permutation.[2] However, that does not necessarily hold, and the public key generated by $\mathcal{R}$ could be anything. In fact, it is possible that the public key generated by the reduction $\mathcal{R}$ is *fake* in the sense that the FDH signatures are no longer unique relative to this fake pk. Once signatures are no longer unique (with respect to the fake pk), it is possible that a *real forger* outputs a forgery σ' on m^* which is different from σ^*, the one provided by reduction $\mathcal{R}$ before the rewind. In fact, it could be possible that $\sigma^* \neq \sigma'$ is no longer useful for $\mathcal{R}$ in order to solve the RSA instance after the rewind, and hence, the impossibility result breaks down.

In Sect. 3, we restate (and prove) a corrected version of Coron's impossibility result. Fortunately, it turns out that Coron's argument can be salvaged by requiring the trapdoor permutation in FDH to be certified. Note that in case of a certified trapdoor permutation it is not possible for the reduction $\mathcal{R}$ to generate a fake public key without detection, and hence, signatures are guaranteed to be unique.

1.3. *A Tight Security Reduction for FDH Signatures*

It is precisely the non-uniqueness of FDH signatures with respect to a fake public key that will allow us to prove a tight security from the lossiness of the TDP [i.e., the Φ-Hiding Assumption in the case of RSA-FDH]. Our proof is surprisingly simple and is sketched as follows. In a first step, we substitute the trapdoor permutation in public key with a lossy one. We use the programmability of the random oracle to show that this remains unnoticed by the adversary assuming lossiness of the TDP. Note that once the TDP is lossy, FDH signatures (i.e., σ with $f(\sigma) = H(m)$) are not longer unique since, by the definition of lossiness, each $H(m)$ has many pre-images under a lossy f. In the second step, we show that any successful forger will be able to find a collision in the TDP, i.e., two values $x \neq \hat{x}$ with $f(x) = f(\hat{x})$, which is again hard assuming lossiness. The full proof is given in Sect. 3.

For the important case of RSA-FDH, this gives a tight security reduction from the Φ-Hiding Assumption, in the random oracle model. The Φ-Hiding Assumption is believed to be true for sufficiently small public RSA exponents $e < N^{1/4-\varepsilon}$ [10]. This in particular includes the important low-exponent cases of $e = 3$ and $e = 2^{16} + 1$ since they allow efficient verification of RSA-FDH signatures.[3]

[2] Such restricted reductions were called *key-preserving reductions* in [27].

[3] We stress that our tight proof technically does not give a counter-example to Coron's impossibility result since our reduction is from the Φ-Hiding Assumption, not the RSA assumption. However, as corollary

It is interesting to remark that, at a conceptual level, FDH is the first signature scheme with *unique signatures* and a *tight* security reduction (from a non-interactive assumption).[4] Previously, only tight security reductions for *randomized* signatures were known (e.g., [4,7,16,18]).

1.4. *The Probabilistic Signature Scheme PSS*

Our observations can also be applied to the probabilistic signature scheme (PSS) [4] which is contained in IEEE P1363a [20], ISO/IEC 9796-2, and PKCS#1 v2.1 [29]. Improving an earlier result by Bellare and Rogaway, [4] Coron proved that, if at least $\log_2(q_s)$ bits of random salt are used in PSS, then there is a tight security reduction from the one-wayness of the TDP [13,14]. Furthermore, Coron also proved that $\log_2(q_s)$ bits of random salt are essentially optimal for a tight security reduction. Our results for PSS are similar to the ones for FDH. We note that Coron's impossibility proof for PSS contains the same gap as the one in FDH, i.e., it is only correct if the underlying trapdoor permutation is certified. We show a tight security proof from lossiness to the security of PSS, with random salt of arbitrary (possibly zero) length.

Our results also apply to PSS with message recovery (PSS-R), where the signature encodes (parts of) the message. We show that PSS-R with arbitrary-length salt is tightly secure assuming lossiness of the TDP. Concretely, our security reduction shows that PSS-R with zero-length salt has an overhead (signature length minus message length) of only $2k$ bits, where k is the security parameter. Interestingly, this matches the overhead of BLS short signatures [8] over bilinear maps.

1.5. *Related Work*

There is a lot of work on FDH and tightly secure signature schemes, we try to summarize part of it relevant to this work.

Tight Security Reduction for RSA-FDH from an Interactive Assumption. Koblitz and Menezes [24, Sec. 3] show a tight reduction from an *interactive assumption* they call the *RSA1 assumption* (which is related to the one-more-RSA assumption RSA-CTI [2]): Given N, e, and a set of $q_s + q_h$ values y_i chosen uniformly from $\mathbb{Z}_N$, the adversary is permitted adaptively to select up to q_s of those y_i for which he is given solutions x_i to $x_i^e = y_i \mod N$. The adversary wins if he produces a solution $x_i^e = y_i \mod N$ for one of the remaining y_i. Even though the RSA1 assumption looks plausible, it is an interactive assumption and almost a tautology for expressing that RSA-FDH signatures are secure in the random oracle model. In fact, our tight security proof for RSA-FDH also serves to show a tight reduction from Φ-Hiding to RSA1.

Non-unique Signatures with Tight Reductions. There exists several previous works that build digital signature schemes with a tight security reduction. We stress that

Footnote 3 Continued
the impossibility result would exclude any (even non-tight) equivalence between the Φ-Hiding and the RSA assumption.

[4]Here, we do not count tight security proofs from "tautological assumptions" which are essentially assuming that the signature scheme is secure.

all of them have, in contrast to FDH, a randomized signing algorithm, i.e., signatures are not unique. Goh et al. [18] show that adding one single bit of random salt to the hash function of FDH allows to prove a tight security reduction from the RSA assumption. Bernstein [7] shows a tight security reduction for (a certain randomized variant of) Rabin–Williams signature scheme from the factoring assumption. More generally, Gentry et al. [16] introduce the concept of pre-image samplable trapdoor functions which are non-injective trapdoor functions with an efficient pre-image sampling algorithm. They further propose a probabilistic variant of FDH and prove it tightly secure. In fact, their proof technique is reminiscent to the second step in our proof of FDH from the lossiness but FDH can not be viewed as an instance of their probabilistic FDH variant.

RSA-OAEP. Recently, [22] used the Φ-Hiding Assumption to show that the RSA function is lossy and used this fact to prove positive instantiability results of RSA-OAEP in the standard model.

1.6. *Open Problems*

On the one hand, the Φ-Hiding Assumption is believed to be true for public exponents $e \leq N^{1/4-\varepsilon}$, and hence, for these values, we get a tight security reduction for RSA-FDH. On the other hand, Coron's impossibility results hold for prime e with $e > N$. This leaves the interesting open problem whether for public exponents $N^{1/4} \leq e \leq N$ there exists a tight security reduction for RSA-FDH (under a reasonable assumption).

2. Definitions

2.1. *Notations and Conventions*

We denote our security parameter as k. For all $n \in \mathbb{N}$, we denote by 1^n the n-bit string of all ones. For any element x in a set S, we use $x \in_R S$ to indicate that we choose x uniformly random in S. All algorithms may be randomized. For any algorithm A, we define $x \leftarrow_\$ A(a_1, \dots, a_n)$ as the execution of A with inputs $a_1, \dots, a_n$ and fresh randomness and then assigning the output to x. We denote the set of prime numbers by $\mathbb{P}$ and we denote the subset of k-bit primes as $\mathbb{P}[k]$. Similarly, we have the integers denoted by $\mathbb{Z}$ and $\mathbb{Z}[k]$. We denote by $\mathbb{Z}_N^*$ the multiplicative group modulo $N \in \mathbb{Z}$.

2.2. *Games*

A game (such as in Fig. 1) is defined as a collection of procedures, as per the model of [5]. There is an **Initialize** procedure and a **Finalize** procedure, as well a procedure for each separate oracle. Executing a game G with and adversary $\mathcal{A}$ means running the adversary and using the procedures to answer any oracle queries. The adversary must first make one query to **Initialize**. Then, it may query the oracles as many times as allowed by the definition of the game. After this, the adversary must then make 1 query to **Finalize**, which is the final procedure call of the game. The output of **Finalize** is denoted by $\mathsf{G}^{\mathcal{A}}$. In particular, we write $\mathsf{G}^{\mathcal{A}} \Rightarrow 0$ and $\mathsf{G}^{\mathcal{A}} \Rightarrow 1$, when **Finalize** outputs 0 respectively 1. Where the **Finalize** procedure simply returns the output of the adversary, we omit the

```
procedure Initialize
$(pk, sk) \leftarrow_{\$} \mathsf{KeyGen}(1^k)$
return $pk$

procedure Hash($m$)
if $(m, \cdot) \in \mathcal{H}$ then fetch $(m, y) \in \mathcal{H}$; return $y$
else $y \in_R \mathsf{Dom}$; $\mathcal{H} \leftarrow \mathcal{H} \cup (m, y)$; return $y$

procedure Sign($m$)                Game UF-CMA
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
return $\sigma \leftarrow_{\$} \mathsf{Sign}(sk, m)$

procedure Finalize($m^*, \sigma^*$)
if $\mathsf{Verify}(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{M}$
then return 1
else return 0
```

Fig. 1. Game defining UF-CMA security in the random oracle model.

Finalize procedure. We use a strongly typed pseudo-code with implicit initialization. Which means all variables maintain their type throughout the execution of the games and they are all implicitly declared and initialized. Boolean flags are initialized to False, numerical types are initialized to 0, and sets are initialized to $\emptyset$.

2.3. *Signature Schemes*

A digital signature is a message-dependant bit string σ, which can only be generated by the signer, using a secret signing key *sk* and is transmitted with the message. The signature can then be verified by the receiver using a public verification key *pk*. A digital signature scheme is defined as a triple of probabilistic algorithms $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, which we describe below:

1. KeyGen takes as an input the unary representation of our security parameter (1^k) and outputs a signing key sk and verification key pk.
2. Sign takes as input a signing key sk, message m and outputs a signature σ.
3. Verify is a deterministic algorithm, which on input of a public key and a message-signature pair (m, σ) outputs 1 (accept) or 0 (reject).

We say that SIG is correct if for all public key and secret key pairs generated by KeyGen, we have:

$$\Pr[\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1] = 1.$$

We now define UF-CMA (unforgeability under chosen message attacks) assuming the signature scheme SIG contains a hash function $h : \{0, 1\}^* \to \mathsf{Dom}$ which is modeled as a random oracle.

We say a signature scheme SIG is $(t, \varepsilon, q_h, q_s)$-UF-CMA secure in the random oracle model, if for all adversaries $\mathcal{A}$ running in time upto t, making at most q_h hashing and q_s signing oracle queries, they have an advantage of at most ε, where the advantage of $\mathcal{A}$ is defined as:

$$\mathbf{Adv}_{\mathsf{SIG}}^{\mathsf{UF\text{-}CMA}}(\mathcal{A}) = \Pr\left[\mathsf{UF\text{-}CMA}^{\mathcal{A}} \Rightarrow 1\right].$$

Here, we assume wlog that $\mathcal{A}$ always makes a query to $\mathsf{Hash}(m)$ before calling $\mathsf{Sign}(m)$ or $\mathsf{Finalize}(m, \cdot)$. What this means is that we always have $q_h > 0$, as at least one hash query must be made before calling Finalize. Furthermore, we see that we have $q_h \geq q_s + 1$, as the adversary must call $\mathsf{Finalize}(\cdot,\cdot)$ with a value not previously submitted to $\mathsf{Sign}(\cdot)$.

2.4. *Trapdoor Permutations*

We recall the definition of trapdoor permutation families.

Definition 1. A family of trapdoor permutations $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ consists of the following three polynomial-time algorithms.

1. The probabilistic algorithm Gen, which on input 1^k outputs a public description *pub* (which includes an efficiently sampleable domain Dom_{pub}) and a trapdoor *td*.
2. The deterministic algorithm Eval, which on input *pub* and $x \in \mathsf{Dom}_{pub}$, outputs $y \in \mathsf{Dom}_{pub}$. We write $f(x) = \mathsf{Eval}(pub, x)$.
3. The deterministic algorithm Invert, which on input *td* and $y \in \mathsf{Dom}_{pub}$, outputs $x \in \mathsf{Dom}_{pub}$. We write $f^{-1}(y) = \mathsf{Invert}(td, y)$.

We require that for all $k \in \mathbb{N}$ and all (pub, td) output by $\mathsf{Gen}(1^k)$, $f(\cdot) = \mathsf{Eval}(pub, \cdot)$ defines a permutation over Dom_{pub} and that for all $x \in \mathsf{Dom}_{pub}$, $\mathsf{Invert}(td, \mathsf{Eval}(pub, x)) = x$.

We want to point out that $f(\cdot) = \mathsf{Eval}(pub, \cdot)$ is only required to be a permutation for correctly generated *pub*, but not every bit string *pub* necessarily yields a permutation. A family of trapdoor permutations TDP is said to be *certified* [6] if the fact that it is a permutation can be verified in polynomial time given *pub*.

Definition 2. A family of trapdoor permutations TDP is called certified if there exists a deterministic polynomial-time algorithm $\mathsf{Certify}$ that, on input of 1^k and an arbitrary (polynomially bounded) bit string *pub* (potentially not generated by Gen), returns 1 iff $f(\cdot) = \mathsf{Eval}(pub, \cdot)$ defines a permutation over Dom_{pub}.

We now recall security notion for trapdoor permutations. A trapdoor permutation TDP is hard to invert (one-way) if given *pub* and $f(x)$ for uniform $x \in \mathsf{Dom}_{pub}$, it is hard to compute x. More formally, it is (t, ε)-hard to invert if for all adversaries $\mathcal{A}$ running in time t, $\Pr[\mathcal{A}(pub, \mathsf{Eval}(pub, x)) = x] \leq \varepsilon$, where the probability is taken over $(pub, td) \leftarrow \mathsf{Gen}(1^k)$, $x \in_R \mathsf{Dom}_{pub}$ and the random coin tosses of $\mathcal{A}$. The following security notion, lossiness [28], is a stronger requirement than one-wayness.

Definition 3. Let $l \geq 2$. A trapdoor permutation TDP is a (l, t, ε) lossy trapdoor permutation if the following two conditions hold.[5]

1. There exists a probabilistic polynomial-time algorithm $\mathsf{LossyGen}$, which on input 1^k outputs pub' such that the range of $f'(\cdot) := \mathsf{Eval}(pub', \cdot)$ under $\mathsf{Dom}_{pub'}$ is at least a factor of l smaller than the domain
$\mathsf{Dom}_{pub'}$: $|\mathsf{Dom}_{pub'}|/|f_{pub'}(\mathsf{Dom}_{pub'})| \geq l$. (Note that we measure the lossiness in its absolute value l, i.e., the function has $\lceil \log_2 l \rceil$ *bits* of lossiness.)

[5] We deviate in two ways from the original definition of lossy trapdoor functions Peikert and Waters [28]. First, we define the permutation over arbitrary domains Dom, rather than $\{0,1\}^k$; second, we measure the absolute lossiness l, rather than the bits of lossiness $\ell = \log_2(l)$.

```
procedure Initialize                         Game $\mathsf{L}_0$
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$
return $pub$

procedure Initialize                         Game $\mathsf{L}_1$
$(pub', \perp) \leftarrow_{\$} \mathsf{LossyGen}(1^k)$
return $pub'$
```

Fig. 2. The lossy trapdoor permutation games.

2. All distinguishers $\mathcal{D}$ running in time at most t have an advantage $\mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D})$ of at most ε (cf. Fig. 2), where:

$$\mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}) = |\Pr[\mathsf{L}_1^{\mathcal{D}} \Rightarrow 1] - \Pr[\mathsf{L}_0^{\mathcal{D}} \Rightarrow 1]|.$$

We say TDP is *regular* (l, t, ε) *lossy* if TDP is (l, t, ε) lossy and all functions $f'(\cdot) = \mathsf{Eval}(pub', \cdot)$ generated by $\mathsf{LossyGen}$ are l-to-1 on $\mathsf{Dom}_{pub'}$.

2.5. *The RSA Trapdoor Permutation*

We define the RSA trapdoor permutation $\mathsf{RSA} = (\mathsf{RSAGen}, \mathsf{RSAEval}, \mathsf{RSAInv})$ as follows. The RSA instance generator $\mathsf{RSAGen}(1^k)$ outputs $pub = (N, e)$ and $td = d$, where $N = pq$ is the product of two random $k/2$-bit primes, e is chosen randomly from the set $\{e |\ \gcd(e, \varphi(N)) = 1\}$, and $d = e^{-1} \mod \varphi(N)$. The domain is $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$. The evaluation algorithm $\mathsf{RSAEval}(pub, x)$ returns $f(x) = x^e \bmod N$, the inversion algorithm $\mathsf{RSAInv}(td, y)$ returns $f_{pub}^{-1}(y) = y^d \bmod N$. The standard assumption is that RSA is hard to invert. We will review the (regular) lossiness of RSA in Sect. 4.

3. Full Domain Hash Signatures

3.1. *The Scheme*

For a familiy of trapdoor permutations $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$, we define the Full Domain Hash ($\mathsf{TDP\text{-}FDH}$) signature scheme [4] in Fig. 3.

```
procedure KeyGen                                         TDP-FDH
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$
Pick a hash function $h : \{0,1\}^* \rightarrow \mathsf{Dom}_{pub}$
return $(pk = (h, pub), sk = td)$

procedure Sign$(sk, m)$
return $\sigma = \mathsf{Invert}(td, h(m))$                  // $\sigma = f^{-1}(h(m))$

procedure Verify$(pk, m, \sigma)$
if $\mathsf{Eval}(pub, \sigma) = h(m)$ then return 1          // $f(\sigma) \overset{?}{=} h(m)$
else return 0
```

Fig. 3. The full domain hash signature scheme $\mathsf{TDP\text{-}FDH}$.

3.2. *Classical Security Results of TDP-FDH*

The original reduction by Bellare and Rogaway from one-wayness of TDP loses a factor of $(q_h + q_s)$ [4], which was later improved by Coron to a factor of q_s [12] for the case of the RSA trapdoor permutation.

Theorem 1. (Coron [12]) *Assume the trapdoor permutation* RSA *is* (t', ε')*-hard to invert. Then, for any* (q_h, q_s), RSA-FDH *is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\varepsilon' = \frac{\varepsilon}{q_s} \cdot \left(1 - \frac{1}{q_s + 1}\right)^{q_s+1} \approx \frac{\varepsilon}{q_s} \cdot \exp(-1)$$
$$t' = t + (q_h + q_s + 1) \cdot \mathcal{O}(k^3).$$

3.3. *A Corrected Version of Coron's Optimality Result*

Coron showed that a security loss of a factor q_s (times some constant) is essentially optimal for TDP-FDH [13,14]. To state a corrected version of Coron's impossibility result, we first recall the following definitions [13].

Definition 4. We say a reduction $\mathcal{R}$ $(t_{\mathcal{F}}, t_{\mathcal{R}}, q_h, q_s, \varepsilon_{\mathcal{F}}, \varepsilon_{\mathcal{R}})$-reduces inverting a trapdoor permutation to breaking SIG = (KeyGen, Sign, Verify) if after running a forger $\mathcal{F}$, in a black-box manner, that $(t_{\mathcal{F}}, q_h, q_s, \varepsilon_{\mathcal{F}})$-breaks SIG, the reduction outputs a pre-image of y, with probability at least $\varepsilon_{\mathcal{R}}$, with running time at most $t_{\mathcal{R}}$.

We now state the corrected version of Coron's impossibility result which we prove in Sect. 5.

Theorem 2. *Suppose* TDP *is a certified trapdoor permutation. Let* $\mathcal{R}$ *be a reduction that* $(t_{\mathcal{F}}, t_{\mathcal{R}}, q_h, q_s, \varepsilon_{\mathcal{F}}, \varepsilon_{\mathcal{R}})$*-reduces breaking one-wayness of* TDP *to breaking* UF-CMA *security of* TDP-FDH. *If* $\mathcal{R}$ *runs the forger only once, then we can build an inverter* $\mathcal{I}$ *which* $(t_{\mathcal{I}}, \varepsilon_{\mathcal{I}})$*-breaks one-wayness of* TDP *with:*

$$t_{\mathcal{I}} \leq 2 \cdot (t_{\mathcal{R}} + t_{\mathcal{F}})$$
$$\varepsilon_{\mathcal{I}} \geq \varepsilon_{\mathcal{R}} - \varepsilon_{\mathcal{F}} \cdot \frac{2 \cdot \exp(-1)}{q_s}.$$

Hence, given a security reduction from one-wayness to the unforgeability of TDP-FDH which loses less than a factor of q_s, one obtains an efficient inverter $\mathcal{I}$ for TDP (with non-negative success probability $\varepsilon_{\mathcal{I}}$). It is worth noting that the success probability of a forger is defined by taking the probability for valid *pk* only.

3.4. *A Tight Security Proof for TDP-FDH*

The impossibility result of Theorem 2 only holds for TDP-FDH if TDP is a certified trapdoor permutation. However, if TDP is not certified, this leaves room for a tight proof

```
procedure Initialize                    Game $\mathsf{G}_0 = (\mathsf{UF\text{-}CMA})$
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$
Return $pk = pub$

procedure Hash$(m)$
if $\mathcal{H}[m]$ is defined then
    fetch $y_m = \mathcal{H}[m]$, return $y_m$
else
    $y_m \in_R \mathsf{Dom}_{pub}$
    $\mathcal{H}[m] := y_m$; return $y_m$

procedure Sign$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
return $\sigma_m = \mathsf{Invert}(td, \text{Hash}(m))$

Procedure Finalize$(m^*, \sigma^*)$
if $(\mathsf{Verify}(pub, m^*, \sigma^*) = 1) \wedge (m^* \notin \mathcal{M})$ return 1
    else return 0
```

```
procedure Initialize                    Games $\mathsf{G}_1$-$\mathsf{G}_4$
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$              $//\mathsf{G}_1, \mathsf{G}_4$
$(pub, \perp) \leftarrow_{\$} \mathsf{LossyGen}(1^k)$      $//\mathsf{G}_2, \mathsf{G}_3$
Return $pk = pub$

procedure Hash$(m)$
if $\mathcal{H}[m]$ is defined then
    fetch $(y_m, \sigma_m) = \mathcal{H}[m]$; return $y_m$
else
    $\sigma_m \in_R \mathsf{Dom}_{pub}$
    $y_m = \mathsf{Eval}(pub, \sigma_m)$
    $\mathcal{H}[m] := (y_m, \sigma_m)$; return $y_m$

procedure Sign$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
call Hash$(m)$
fetch $(y_m, \sigma_m) = \mathcal{H}[m]$, return $\sigma_m$

Procedure Finalize$(m^*, \sigma^*)$
call Hash$(m^*)$, fetch $(y_{m^*}, \sigma_{m^*}) = \mathcal{H}[m^*]$ $//\mathsf{G}_3, \mathsf{G}_4$
if $\sigma_{m^*} = \sigma^*$ then BAD = true return 0 $//\mathsf{G}_3, \mathsf{G}_4$
if $\mathsf{Verify}(pub, m^*, \sigma^*) = 1 \wedge (m^* \notin \mathcal{M})$ return 1
    else return 0
```

Fig. 4. Games for the proof of Theorem 3.

for TDP-FDH. We now state our main result, namely that TDP-FDH is tightly secure assuming TDP, is regular lossy.

Theorem 3. *Assume* TDP = (Gen, Eval, Invert) *is a regular* (l, t', ε')*-lossy trapdoor permutation for* $l \geq 2$*. Then, for any* (q_h, q_s), TDP-FDH *is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\varepsilon = \left(\frac{2l-1}{l-1}\right) \cdot \varepsilon'$$
$$t = t' - (q_h + q_s + 1) \cdot T_{\mathsf{TDP}}$$

and T_{TDP} *is the time to evaluate* TDP.

Proof. We prove our theorem using a series of games. The description of these games is found in Fig. 4. Let $\mathcal{A}$ be an adversary that runs in time t against TDP-FDH executed in the UF-CMA experiment described in G_0 in Fig. 1 with $\varepsilon = \Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1]$. Recall that we assume wlog that $\mathcal{A}$ always makes a query to $\mathsf{Hash}(m)$ before calling $\mathsf{Sign}(m)$ or $\mathsf{Finalize}(m, \cdot)$.

Lemma 1. $\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]$.

Proof. In G_0, we modeled the hash function as a random oracle. In G_1 we modify the random oracle and the signing queries. On any m the random oracle now works by evaluating the permutation on a random element $\sigma_m \in \mathsf{Dom}_{pub}$. We then modify the signing oracle to return this element σ_m. Note that signing no longer requires the trapdoor td. It can be seen that all our signatures will verify due to the fact that $\mathsf{Eval}(pub, \sigma_m) = y_m$ for all m. Thus, our simulation of the signatures is correct. Since TDP is a permutation, the distribution of our hash queries in G_1 is identical to the distribution in G_0. Thus, we have $\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]$. □

Lemma 2. *There exists a distinguisher $\mathcal{D}_1$ against the lossiness of* TDP*, which runs in time $t = t_{\mathcal{A}} + (q_h + q_s) \cdot T_{\mathsf{TDP}}$ and such that* $|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]| = \boldsymbol{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1)$.

Proof. From G_1 to G_2, we change the key generation from a normal permutation to a lossy permutation; however, the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_1$ against the lossiness of TDP, using these games. The distinguisher will run $\mathcal{A}$ and simulates the oracles $\mathsf{Sign}(\cdot)$, $\mathsf{Hash}(\cdot)$ as described in games G_1&G_2, for which it requires time $(q_h + q_s) \cdot T_{\mathsf{TDP}}$. Note that $\mathcal{D}_1$ does not require the trapdoor td to simulate the oracles. After $\mathcal{A}$ calls $\mathsf{Finalize}$, $\mathcal{D}_1$ returns the output of $\mathsf{Finalize}$. Thus, we can see that $\Pr[\mathrm{L}_0^{\mathcal{D}_1} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]$. Similarly, we have $\Pr[\mathrm{L}_1^{\mathcal{D}_1} \Rightarrow 1] = \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$. Hence, we have $|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]| = |\Pr[\mathrm{L}_0^{\mathcal{D}_1} \Rightarrow 1] - \Pr[\mathrm{L}_1^{\mathcal{D}_1} \Rightarrow 1]| = \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1)$. □

Lemma 3. $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \left(\frac{l-1}{l}\right)\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$.

Proof. In G_3, we introduce a new rule, which sets BAD to true if the forgery σ^* provided by $\mathcal{A}$ is the same as the simulated signature σ_{m^*} for the target message m^*. If this is the case, the adversary loses the game, i.e., G_3 outputs 0. σ_{m^*} is independent of $\mathcal{A}$'s view and is uniformly distributed in the set of pre-images of y_{m^*}. Due to the l regular lossiness of TDP, the probability of a collision is equal to exactly $1/l$. Thus, we see that the BAD rule reduces the probability of the adversary winning the game by $1/l$, hence $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = (1 - \frac{1}{l})\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] = \left(\frac{l-1}{l}\right)\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$. □

Lemma 4. *There exists a distinguisher $\mathcal{D}_2$ against the lossiness of* TDP*, which runs in time $t = t_{\mathcal{A}} + (q_h + q_s) \cdot T_{\mathsf{TDP}}$ and that* $|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]| = \boldsymbol{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2)$.

Proof. From G_3 to G_4, we change the key generation from a lossy permutation to a normal permutation; however, the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_2$ against the lossiness of TDP, using these games. The distinguisher will act as the challenger to $\mathcal{A}$. It will simulate the oracles as described in games G_3&G_4, for which it requires time $(q_h + q_s) \cdot T_{\mathsf{TDP}}$. After $\mathcal{A}$ calls $\mathsf{Finalize}$, $\mathcal{D}_2$ returns the output of $\mathsf{Finalize}$. We can see that $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{L}_0^{\mathcal{D}_2} \Rightarrow 1]$. Similarly, we have $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{L}_1^{\mathcal{D}_2} \Rightarrow 1]$. Hence, we have $|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]| = |\Pr[\mathrm{L}_1^{\mathcal{D}_2} \Rightarrow 1] - \Pr[\mathrm{L}_0^{\mathcal{D}_2} \Rightarrow 1]| = \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2)$. □

Lemma 5. $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = 0$.

Proof. In G_4 we again use the original KeyGen such that $\mathsf{Eval}(pub, \cdot)$ defines a permutation. This means that our signing function is now a permutation, and thus, any forgery implies a collision. Therefore, whenever the adversary is able to make a forgery, the game outputs 0 due to the BAD rule. Whenever they are unable to make a forgery, the game outputs 0. Thus, we can see that in all cases, the game will output 0, hence $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = 0$. □

We combine Lemmas 1 to 5 to get:

$$\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1) + \left(\frac{l}{l-1}\right)\mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2).$$

where l is the lossiness of TDP. Because the distinguishers run in the same time, we know that both distinguishers can have at most an advantage of ε', giving us:

$$\varepsilon \leq \frac{2l-1}{l-1} \cdot \varepsilon'.$$

This completes the proof. □

4. Lossiness of RSA from the Φ-Hiding Assumption

4.1. *Lossiness of RSA*

The lossiness of RSA for a number of specific instance generators RSAGen was first considered in [22]. We now recall (and extend) some of the results from [22].

First, we recall some definitions from [22]. We denote by $\mathcal{RSA}_k := \{(N, p, q) \mid N = pq, p, q \in \mathbb{P}_{k/2}, p \neq q\}$ the set of all the tuples (N, p, q) such that $N = pq$ is the product of two distinct $k/2$-bit primes. Such an N is called an RSA modulus. By $(N, p, q) \in_R \mathcal{RSA}_k$ we mean the (N, p, q) is sampled according to the uniform distribution on $\mathcal{RSA}_k$. Let R be some relation on p and q. By $\mathcal{RSA}_k[R]$, we denote the subset of $\mathcal{RSA}_k$ such that the relation R holds on p and q. For example, let e be a prime. Then $\mathcal{RSA}_k[p = 1 \bmod e]$ is the set of all (N, p, q), where $N = pq$ is the product of two distinct $k/2$-bit primes p, q and $p = 1 \bmod e$. That is, the relation $R(p, q)$ is true if $p = 1 \bmod e$ and q is arbitrary. By $(N, p, q) \in_R \mathcal{RSA}_k[R]$ we mean that (N, p, q) is sampled according to the uniform distribution on $\mathcal{RSA}_k[R]$.

α-Φ-Hiding Assumption. We recall a variant of the Φ-Hiding Assumption introduced by Cachin, Micali and Stadler [10], where we build on a formalization by Kiltz, O'Neil and Smith [22]. The main statement of the assumption is that given an k-bit RSA modulus $N = pq$ and a random $\alpha \cdot k$-bit prime e (where $0 < \alpha < \frac{1}{4}$ is a public constant such that $\alpha \cdot k \in \mathbb{Z}$), it is difficult to decide if $e \mid \varphi(N)$ or if $gcd(e, \varphi(N)) = 1$. We note that if $e \mid \varphi(N)$ with $e \geq N^{1/4}$, then N can be factored using Coppersmith's attacks [11], see [10] for details. Hence, for the α-Φ-Hiding Assumption to hold, the bit length of e must not exceed one-fourth of the bit length of N.

procedure Initialize Game P_0	**procedure Initialize** Game P_1
$e \in_R \mathbb{P}[\alpha k]$	$e \in_R \mathbb{P}[\alpha k]$
$R = (gcd(e, \varphi(N)) = 1, N \mod e \neq 1)$	$R = (p = 1 \mod e, p \neq 1 \mod e^2, q \neq 1 \mod e)$
$(N, p, q) \in_R \mathcal{RSA}_k[R]$	$(N, p, q) \in_R \mathcal{RSA}_k[R]$
return (N, e)	return (N, e)

Fig. 5. The α-Φ-Hiding Assumption Games.

Consider a distinguisher $\mathcal{D}$ which plays one of the games P_0 or P_1 defined in Fig. 5. The advantage of $\mathcal{D}$ is defined as:

$$\mathbf{Adv}^{\Phi\mathrm{H}}(\mathcal{D}) = |\Pr[\mathsf{P}_1^{\mathcal{D}} \Rightarrow 1] - \Pr[\mathsf{P}_0^{\mathcal{D}} \Rightarrow 1]|.$$

We say that the α-Φ-Hiding Problem is (t, ε)-hard if for all distinguishers $\mathcal{D}$ running in time at most t have an advantage of at most ε.

Define an RSA instance generator RSAGen as an algorithm that returns (N, e, p, q) sampled as $e \in_R \mathbb{P}_{\alpha k}$ and $(N, p, q) \in_R (gcd(e, \varphi(N)) = 1, N \mod e \neq 1)$. (See [22] for details on the sampling algorithm.)

Lemma 6. *If the α-Φ-Hiding Problem is (t, ε)-hard, then the* $\mathsf{RSA} = (\mathsf{RSAGen}, \mathsf{RSAEval}, \mathsf{RSAInv})$ *defines a $(2^\alpha, t, \varepsilon)$-lossy trapdoor permutation.*

Proof. If (N, e) is sampled using RSAGen, then $\gcd(e, \varphi(N) = 1)$ and (N, e) defines a permutation $\mathsf{RSA}(x) = x^e \mod N$ over $\mathbb{Z}_N^*$. We define $\mathsf{LossyGen}$ to be an algorithm that returns (N, e) sampled as $e \in_R \mathbb{P}[\alpha k]$ and $(N, p, q) \in_R (p = 1 \mod e, p \neq 1 \mod e^2, q \neq 1 \mod e)$. If (N, e) is sampled using $\mathsf{LossyGen}$ then $e \mid \varphi(N)$, and hence, the RSA function is exactly e-to-1 on the domain $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$. We note in particular that $e^2 \not| (p - 1)$, and thus, we have exactly an e-to-1 function. By definition, the outputs of RSAGen and $\mathsf{LossyGen}$ are indistinguishable if the α-Φ-Hiding Problem is hard. □

Remark 1. We use the classical definition of lossiness here, due to the fact that our public exponent e is not a parameter, and thus, we cannot give the exact regular lossiness of the RSA trapdoor permutation.

FIXED-PRIME Φ-HIDING ASSUMPTION. In practice, e is chosen to be small and is generally fixed to some specific numbers, such as $e = 3$ or $e = 2^{16} + 1$, which allows for fast exponentiation. We now show a minor variant of the α-Φ-Hiding Assumption for *fixed primes e*, where our formalization relies on discussions from [10] and [22, Footnote 9].

First, we discuss the special case of $e = 3$. We define our RSA instance RSAGen_3 generator as an algorithm that samples (N, p, q) uniformly from $\mathcal{RSA}_k[p = 2 \mod 3, q = 2 \mod 3]$, which is equivalent to $\mathcal{RSA}_k[gcd(3, \varphi(N)) = 1]$. We note that $N \mod 3$ is always 1. This means that for the lossy case, we must also ensure the $N \mod 3 = 1$, otherwise there would be a simple distinguisher. To ensure this is to have 3 divide both $p - 1$ and $q - 1$. Thus, our lossy keys are sampled from the $(p, q = 1 \mod 3, p \neq 1 \mod 9, q \neq 1 \mod 9)$.

procedure Initialize Game $3\mathsf{F}_0$	**procedure Initialize** Game $3\mathsf{F}_1$
$R = (gcd(3, \varphi(N)) = 1)$	$R = (p, q = 1 \mod 3, p \neq 1 \mod 9, q \neq 1 \mod 9)$
$(N, p, q) \in_R \mathcal{RSA}_k[R]$	$(N, p, q) \in_R \mathcal{RSA}_k[R]$
return $(N, e = 3)$	return $(N, e = 3)$

Fig. 6. The Fixed-Prime Φ-Hiding Assumption Games.

procedure Initialize Game F_0	**procedure Initialize** Game F_1
$R = (gcd(e, \varphi(N)) = 1, N \mod e \neq 1)$	$R = (p = 1 \mod e, p \neq 1 \mod e^2, q \neq 1 \mod e)$
$(N, p, q) \in_R \mathcal{RSA}_k[R]$	$(N, p, q) \in_R \mathcal{RSA}_k[R]$
return (N, e)	return (N, e)

Fig. 7. The Fixed-Prime Φ-Hiding Assumption Games.

Consider a distinguisher $\mathcal{D}$ which plays one of the games in Fig. 6. The advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\mathrm{F}\Phi\mathrm{H}}(\mathcal{D}) = \left| \Pr[3\mathsf{F}_1^{\mathcal{D}} \Rightarrow 1] - \Pr[3\mathsf{F}_0^{\mathcal{D}} \Rightarrow 1] \right|.$$

We say that the Fixed-Prime Φ-Hiding Problem, with $e = 3$, is (t, ε)-hard if all distinguishers running in time at most t have an advantage of at most ε.

Lemma 7. *If the Fixed-Prime Φ-Hiding Problem, with $e = 3$, is (t, ε)-hard, then the* $\mathsf{RSA}_3 = (\mathsf{RSAGen}_3, \mathsf{RSAEval}, \mathsf{RSAInv})$ *defines a regular $(9, t, \varepsilon)$-lossy trapdoor permutation.*

Proof. If $(N, p, q) \in (gcd(3, \varphi(N)) = 1)$, then $(N, 3)$ clearly makes the RSA function a permutation. If $(N, p, q) \in (p, q = 1 \mod 3, p \neq 1 \mod 9, q \neq 1 \mod 9)$, then $9 \mid \varphi(N)$ and, hence, the RSA function is 9-to-1 on the domain $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$. $\square$

We now consider the general case of fixed $e > 3$. For this case, we define our RSA instance generator RSAGen_e as an algorithm that samples (N, p, q) from $(gcd(e, \varphi(N)) = 1, N \mod e \neq 1)$. We note that $N \mod e$ will be some value between 2 and $e - 1$. This means that for the lossy case, we require e to divide $p - 1$ and not $q - 1$, otherwise we would have a simple distinguisher. Our lossy keys are sampled from $(p = 1 \mod e, p \neq 1 \mod e^2, q \neq 1 \mod e)$.

Consider a distinguisher $\mathcal{D}$ which plays one of the games in Fig. 7. The advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\mathrm{F}\Phi\mathrm{H}}(\mathcal{D}) = | \Pr[\mathsf{F}_1^{\mathcal{D}} \Rightarrow 1] - \Pr[\mathsf{F}_0^{\mathcal{D}} \Rightarrow 1]|.$$

We say that the Fixed-Prime Φ-Hiding Problem, with $e > 3$, is (t, ε)-hard if for all distinguishers running in time at most t have an advantage of at most ε.

Lemma 8. *If the Fixed-Prime Φ-Hiding Problem, with $e > 3$, is (t, ε)-hard, then* $\mathsf{RSA}_e = (\mathsf{RSAGen}_e, \mathsf{RSAEval}, \mathsf{RSAInv})$ *defines a regular (e, t, ε)-lossy trapdoor permutation.*

Proof. If $(N, p, q) \in (gcd(e, \varphi(N)) = 1, N \mod e \neq 1)$, then (N, e) clearly defines a permutation. If $(N, p, q) \in (p = 1 \mod e, p \neq 1 \mod e^2, q \neq 1 \mod e)$, then $e \mid \varphi(N)$, and hence, the RSA function is e-to-1 on the domain $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$. We note in particular that $e^2 \nmid (p-1)$; thus, we have exactly an e-to-1 function. □

5. Proof of Coron's Impossibility Results (Corrected)

5.1. *Meta-reductions*

We now proceed to present a corrected version of Coron's proofs. These proofs use a technique know as a meta-reduction, in which we perform a proof by reduction on a extant reduction. In a normal setting, a reduction $\mathcal{R}$ would interact with an adversary $\mathcal{A}$ in a somewhat black-box manner. While $\mathcal{R}$ is unable to modify or even view the internal state of $\mathcal{A}$, it is still able to rewind $\mathcal{A}$ to a previous state. In particular, this means that $\mathcal{R}$ can rewind the randomness tape of $\mathcal{A}$.

On the other hand, the reduction $\mathcal{R}$ normally interacts with a challenger $\mathcal{C}$ for some (non-interactive) hard problem. After having received an input from $\mathcal{C}$ and possibly some pre-computations, $\mathcal{R}$ begins to interact with $\mathcal{A}$. At some point, $\mathcal{A}$ gives some final output to $\mathcal{R}$, which $\mathcal{R}$ uses to compute a solution for $\mathcal{C}$. The reduction then sends this solution to $\mathcal{C}$. We represent this pictorially in Fig. 8.

The meta-reduction $\mathcal{M}$'s interaction with $\mathcal{R}$ will be twofold. In the first instance, $\mathcal{M}$ will simulate an adversary $\mathcal{A}'$ for $\mathcal{R}$ to interact with as described above. This means that $\mathcal{A}'$ will simulate all the queries of $\mathcal{A}$ and will behave in an expected manner when having its state and/or randomness tape rewound. In the second instance, $\mathcal{M}$ will not only simulate a challenger $\mathcal{C}'$, but will also interact with $\mathcal{R}$ in the same way that $\mathcal{R}$ interacted with $\mathcal{A}$. That is to say $\mathcal{M}$ can rewind $\mathcal{R}$ to a previous state, which includes rewinding $\mathcal{R}$'s randomness tape. We represent this pictorially in Fig. 9.

It is worth noting at this juncture that the running times of $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{M}$ do not consider the running time of the other algorithms that they are running with. What this means is that we assume that an query to another algorithm is answered instantly, that is to say, in 1 time unit. We do this in order to eliminate any "waiting time", wherein an algorithm is waiting for a response from another algorithm, which is superfluous for our analysis.

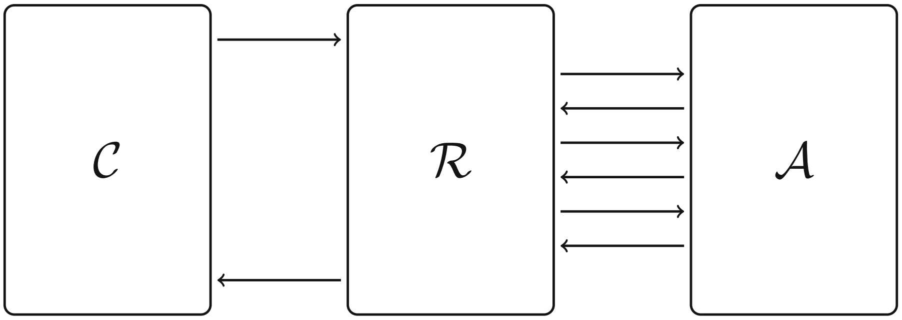

Fig. 8. Visualization of a reduction.

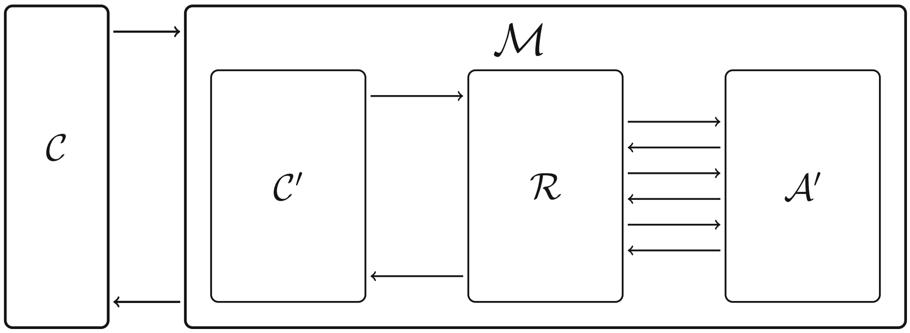

Fig. 9. Visualization of a meta-reduction.

5.2. *Proof of Theorem* 2

Proof. Assume $\mathcal{R}$ is a reduction that $(t_{\mathcal{F}}, t_{\mathcal{R}}, q_h, q_s, \varepsilon_{\mathcal{R}}, \varepsilon_{\mathcal{F}})$-reduces inverting TDP to breaking $\mathsf{TDP\text{-}FDH}$. We will now build an adversary $\mathcal{I}$ against the one-wayness of TDP. $\mathcal{I}$ receives *pub* and $y = \mathsf{Eval}(pub, x) \in \mathsf{Dom}_{pub'}$ for unknown $x \in_R \mathsf{Dom}_{pub}$. The goal of $\mathcal{I}$ is to compute x.

1. Adversary $\mathcal{I}$ runs the reduction $\mathcal{R}$ providing it with (pub, y) and in reply receives a public key $pk = pub'$ for $\mathsf{TDP\text{-}FDH}$. Note that pub' provided by $\mathcal{R}$ may be different from *pub*. $\mathcal{I}$ verifies that pub' defines a permutation on $\mathsf{Dom}_{pub'}$ by running $\mathsf{Certify}(1^k, pub')$. If not, then $\mathcal{I}$ outputs $\perp$ to $\mathcal{R}$. Otherwise, $\mathcal{I}$ continues by simulating a forger $\mathcal{F}_{\text{sim}}$ for $\mathcal{R}$. Recall that the success probability of a forger is defined for valid *pk* only, hence we abort if we receive an invalid *pk*. (We note that it may be the case that $pub' \neq pub$, in particular we do not know if pub' was correctly generated but this does not matter for what follows.)
2. Let $q = \max(q_h, 2\cdot q_s)$. The adversary $\mathcal{I}$ picks a set $\mathcal{X}$ of q arbitrary messages (e.g., at random or the lexicographically first). $\mathcal{I}$ then picks $i \in_R \{1, \ldots, q_s\}$, $m^* \in_R \mathcal{X}$ and $(m_1, \ldots, m_{q_s}) \in_R (\mathcal{X}\setminus\{m^*\})^{q_s}$. This defines the following two sequences of messages:

$$\mathcal{M}^{\text{rw}} = (m_1, \ldots, m_{q_s}), \quad \mathcal{M}^{\text{first}} = (m_1, \ldots, m_{i-1}, m^*).$$

3. Adversary $\mathcal{I}$ queries the signing oracle, with implicit hash queries, on the messages in $\mathcal{M}^{\text{first}}$ (i.e., $(m_1, \ldots, m_{i-1}, m^*)$) and receives the response list $\mathcal{S}^{\text{first}} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma^*)$, if the reduction does not abort. We stress that these are all valid signatures, which we can check using the public verification key. If any of the signatures is not valid, and the reduction does not abort, then we would abort.
4. Adversary $\mathcal{R}$ is then rewound back to its initial state.[6] Now $\mathcal{I}$ queries the signing oracle on the messages $\mathcal{M}^{\text{rw}}$ (i.e., $(m_1, \ldots, m_{q_s})$) and receives $\mathcal{S}^{\text{rw}} = (\sigma_1, \ldots, \sigma_{q_s})$, if the reduction does not abort. We stress that these are all valid signatures, which

[6]More formally, $\mathcal{R}$ is run again with the same random tape. Hence, upto this point $\mathcal{R}$ behaves exactly the same as in its first execution.

we can check using the public verification key. If any of the signatures is not valid, and the reduction does not abort, then we would abort.

5. Adversary $\mathcal{I}$ then makes a hash query on m^* and $q_h - q_s - 1$ additional messages from the set $\mathcal{X}$. This means that $\mathcal{I}$ has made exactly q_h hash queries, including the implicit hash queries from signing, so that it matches what the reduction expects.
6. Adversary $\mathcal{I}$ then tosses a biased coin τ with probability $\varepsilon_{\mathcal{F}}$ of returning 1, and if $\tau = 0$, then $\mathcal{I}$ sends $\perp$ to $\mathcal{R}$. If $\tau = 1$, then $\mathcal{I}$ submits (m^*, σ^*) as a forgery. Note that this is done in time $t_{\mathcal{F}}$ in order to correctly simulate a forger.
7. Because the reduction was rewound, this constitutes a valid forgery, as m^* was not queried to the signature oracle and σ^* is indeed a valid signature on m^*. $\mathcal{R}$ will then return x (with probability $\varepsilon_{\mathcal{R}}$) which $\mathcal{I}$ submits as its solution to one-way experiment.

We now analyze $\mathcal{I}$'s success probability in breaking one-wayness of TDP. To this end we define $\mathcal{Q}$ as the set of all sequences of indices such that the corresponding signature queries are correctly answered by $\mathcal{R}$, after the hash queries (in time less than $t_{\mathcal{R}}$). If a sequence of signature queries is correctly answered by $\mathcal{R}$, then also the same sequence of signature queries without the last message is correctly answered by $\mathcal{R}$, so for any sequence $(m_1, \ldots, m_j) \in \mathcal{Q}$, we have $(m_1, \ldots, m_{j-1}) \in \mathcal{Q}$.

Consider (as a thought-experiment) a (possibly non-efficient) real forger $\mathcal{F}_{\text{real}}$ who inputs arbitrary pub' such that $\mathsf{Certify}(pub', 1^k) = 1$, makes hash queries to the messages from $\mathcal{U}$ (receiving answers $\mathcal{H}$), signature queries to the messages from sequence $\mathcal{M}^{\text{rw}}$ (receiving answers $\mathcal{S}^{\text{rw}}$) and outputs a valid forgery σ^* on the message m^* with probability $\varepsilon_{\mathcal{F}_{\text{real}}}$. This is a valid forger so by the assertion of the theorem reduction $\mathcal{R}$ (interacting with $\mathcal{F}_{\text{real}}$) outputs $x = \mathsf{Invert}(pub, y)$ with probability at least $\varepsilon_{\mathcal{R}}$. After the rewind, $\mathcal{R}$ (interacting with $\mathcal{I}$) sees exactly the same transcript as he would interact with $\mathcal{F}_{\text{real}}$, except if $\mathcal{M}^{\text{first}} \notin \mathcal{Q}$ ($\mathcal{R}$ does not answer all the signature queries before the rewind) and $\mathcal{M}^{\text{rw}} \in \mathcal{Q}$ ($\mathcal{R}$ answers all the signature queries after the rewind). In that case the forger $\mathcal{F}_{\text{real}}$ would output a valid forgery (with probability $\varepsilon_{\mathcal{F}_{\text{real}}}$) but our simulated forger does not. Here, we are relying on the fact that $\mathsf{Eval}(pub', \cdot)$ is a permutation, and hence, the forgery σ^* on m^* output by $\mathcal{F}_{\text{real}}$ is the same as the simulated one. (Otherwise it could be the case that the simulated forgery is useless for the reduction.)

Let $\mathcal{R}^{\mathcal{F}_{\text{real}}}$ denote the execution of $\mathcal{R}$ with the above real forger and let $\mathcal{R}^{\mathcal{F}_{\text{sim}}}$ denote the execution of $\mathcal{R}$ with the forger $\mathcal{F}_{\text{sim}}$ simulated by $\mathcal{I}$. The two games $\mathcal{R}^{\mathcal{F}_{\text{real}}}$ and $\mathcal{R}^{\mathcal{F}_{\text{sim}}}$ are identical unless $\mathcal{M}^{\text{rw}} \in \mathcal{Q}$ and $\mathcal{M}^{\text{first}} \notin \mathcal{Q}$, and $\tau = 1$, where all three "bad events" are defined in the game involving inverter $\mathcal{I}$. By the above we get

$$|\Pr[\mathcal{R}^{\mathcal{F}_{\text{sim}}}(pub, y) = x] - \Pr[\mathcal{R}^{\mathcal{F}_{\text{real}}}(pub, y) = x]| \le \varepsilon_{\mathcal{F}} \cdot \Pr[\mathcal{M}^{\text{rw}} \in \mathcal{Q} \wedge \mathcal{M}^{\text{first}} \notin \mathcal{Q}]$$

We need the following combinatorial lemma due to Coron [13, Appendix D].

Lemma 9. *Let $\mathcal{Q}$ be a set of sequences of at most q_s integers in $\mathcal{X}$, such that for any sequence $(m_1, \ldots, m_j) \in \mathcal{Q}$, we have $(m_1, \ldots, m_{j-1}) \in \mathcal{Q}$. Then:*

$$\Pr_{\substack{i \in_R \{1,\ldots,q_s\} \\ (m_1,\ldots,m_{q_s},m^*) \in_R \mathcal{X}^{q_s+1}}} [(m_1, \ldots, m_{q_s}) \in \mathcal{Q} \wedge (m_1, \ldots, m_{i-1}, m^*) \notin \mathcal{Q}] \le \frac{\exp(-1)}{q_s}.$$

By Lemma 9 we have

$$\Pr[\mathcal{M}^{\mathrm{rw}} \in \mathcal{Q} \wedge \mathcal{M}^{\mathrm{first}} \notin \mathcal{Q}] \leq \frac{\exp(-1)}{q_s}\left(1 - \frac{q_s}{q}\right)^{-1}.$$

(We note that our m_i are drawn from $\mathcal{X}\backslash\{m^*\}$ instead of $\mathcal{X}$, as stated in the Lemma. This leads to the term $\left(1 - \frac{q_s}{|\mathcal{X}|}\right)^{-1}$, which is the bound on the probability that $m_i \neq m^*, \forall i\{1, \ldots, q_s\}$, which then allows us to use the Lemma.) Overall, we obtain for the success probability $\varepsilon_{\mathcal{I}}$ of $\mathcal{I}$:

$$\begin{aligned}\varepsilon_{\mathcal{I}} &= \Pr[\mathcal{R}^{\mathcal{F}_{\mathrm{sim}}}(pub, y) = x] \\ &\geq \Pr[\mathcal{R}^{\mathcal{F}_{\mathrm{real}}}(pub, y) = x] - \varepsilon_{\mathcal{F}} \cdot \frac{\exp(-1)}{q_s}\left(1 - \frac{q_s}{q}\right)^{-1} \\ &\geq \varepsilon_{\mathcal{R}} - \cdot \varepsilon_{\mathcal{F}} \cdot \frac{2 \cdot \exp(-1)}{q_s}.\end{aligned}$$

The last inequality is due to the fact the $q \geq 2 \cdot q_s$, which gives $\left(1 - \frac{q_s}{q}\right)^{-1} \leq 2$. The time bound comes from running the reduction once, plus the rewind and the additional time for at most q_s additional signing queries and the rewind. This is essentially equal to running the reduction twice and simulating the forger twice. □

5.3. *Impossibility for Any Certified Unique Signature Scheme*

Coron also showed that an analogue of this result can be shown to hold in the standard model [13]. This result gives essentially the same bound, with the number of hash queries q_h replaced by the size of the message space 2^{ℓ}. Above and beyond moving the standard model, we further extend Theorem 2 in two directions, simultaneously. Firstly, we extend it to reductions from any non-interactive hard problem Π to forging any certified unique signature scheme SIG. Secondly we allow the reduction to run the forger $\mathcal{F}$ multiple times.

We first define a unique signature scheme and then we define a certified unique signature scheme.

Definition 5. A signature scheme SIG is said to be a unique signature scheme if for all $(pk, sk) \leftarrow_{\$}$ KeyGen, for any message m, there exists exactly one valid signature σ such that $\mathsf{Verify}(pk, m, \sigma_m) = 1$.

Definition 6. A signature scheme SIG is said to be a certified unique signature scheme if it is a unique signature scheme and there exists a polynomial-time algorithm Certify, that on input 1^k and a public key pk, which may have been generated adversarially, will output 1 iff pk defines a unique signature scheme.

We will now recall the definition of a non-interactive problem instance generator due to Abe et al. [1]. This definition is very generic and encompasses both search and decisional problems.

Definition 7. A non-interactive problem instance generator consists of a triple of PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{Verify}, \mathsf{U})$ such that:

- $\mathsf{Gen}(1^k)$ outputs an instance I and a witness w.
- $\mathsf{Verify}(I, S, w)$, where S is a candidate solution and outputs 1 or 0 that represents acceptance or rejection, respectively.
- $\mathsf{U}(I)$ outputs a candidate solution S.

U can be seen as a trivial guessing algorithm against which the advantage of an adversary $\mathcal{A}$ is measured. Concretely, we define the advantage of an adversary $\mathcal{A}$ as:

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathsf{NIP}}(\mathcal{A}) = \Pr[(I, w) \leftarrow_{\$} \mathsf{Gen}(1^k), S \leftarrow_{\$} \mathcal{A}(I) : \mathsf{Verify}(I, S, w) \to 1] \\ - \Pr[(I, w) \leftarrow_{\$} \mathsf{Gen}(1^k), S \leftarrow_{\$} \mathsf{U}(I) : \mathsf{Verify}(I, S, w) \to 1].\end{aligned}$$

We say that Π is (t, ε)-hard if for all PPT adversaries $\mathcal{A}$ running in time at most t have and $\mathbf{Adv}_{\Pi}^{\mathsf{NIP}}(\mathcal{A}) \leq \varepsilon$.

Examples of Non-interactive Problems. We now give a description of some known problems, both search and decisional, in terms of non-interactive problems. Firstly, we have the RSA problem which states that it is hard to invert the RSA trapdoor permutation. An instance of the RSA problem is given as $I = (N, e, y = x^e \mod N)$, where (N, e) are taken from the output of RSAGen, as defined in Sect. 4.1, $x \in_R \mathbb{Z}_N$, and the witness is $w = x$. If the RSA problem is (t, ε)-hard, then this defines a (t, ε)-hard non-interactive problem, where the trivial guessing algorithm U simply picks a random element in $\mathbb{Z}_N^*$. Secondly, we consider the Φ-Hiding Problem, as described in Sect. 4.1. An instance of this problem is given as $I = (N, e)$ and the witness is $w = 1$ if $\gcd(e, \varphi(N)) = 1$ and $s = 0$ if $e \mid \varphi(N)$. The Verify algorithm takes as input (I, b, s) and outputs 1 if $b = s$, else 0. Hence, if the Φ-Hiding Problem is (t, ε)-hard, then this defines a (t, ε)-hard problem with U picking a random bit. Note that we can also define the Computational and Decisional Diffie–Hellman problems in terms of a hard non-interactive problem. Furthermore, this definition also covers problems with non-unique solutions, such as computing modular square roots or approximate-SVP. It is due to these types of problem that we require the Verify function. In these cases, we may not be able to store all the possible solutions, but we can verify a solution using the witness.

We are now ready to state our main impossibility result which generalizes Theorem 2.

Theorem 4. *Suppose* SIG *is a certified unique signature scheme, where the size of the message space is at least* 2^ℓ. *Let* $\mathcal{R}$ *be a reduction that* $(t_{\mathcal{F}}, t_{\mathcal{R}}, q_s, \varepsilon_{\mathcal{F}}, \varepsilon_{\mathcal{R}}, r)$*-reduces breaking a non-interactive problem* Π *to breaking UF-CMA security of* SIG, *where* $\mathcal{R}$ *can run the forger at most* r *times sequentially. Then we can build an adversary* $\mathcal{A}$ *which* $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$*-breaks* Π *with*

$$\begin{aligned}t_{\mathcal{A}} &\leq 2r \cdot (t_{\mathcal{R}} + t_{\mathcal{F}}) \\ \varepsilon_{\mathcal{A}} &\geq \varepsilon_{\mathcal{R}} - \varepsilon_{\mathcal{F}} \cdot \frac{\exp(-1) \cdot r}{q_s} \left(1 - \frac{q_s}{2^\ell}\right)^{-1}.\end{aligned}$$

Proof. The proof or this theorem is very similar to that of Theorem 2. For $1 \leq n \leq r$, we say that the reduction $\mathcal{R}$ is in the nth round when it has already run the forger $n-1$ times. We use this reduction to construct and adversary $\mathcal{A}$ against the problem Π. The reduction is initialized by the adversary $\mathcal{A}$ sending it and instance I of the problem Π. We consider the nth round of the reduction below:

1. The reduction $\mathcal{R}$ provides the adversary $\mathcal{A}$ with a public key pk_n. Note that this public key may be different to the public keys for the previous rounds. $\mathcal{A}$ verifies that pk_n defines a unique signature scheme by running $\mathsf{SIG.Certify}(1^k, pk_n)$. Recall that the success probability of a forger is defined for valid pk only, hence we abort if we receive an invalid pk. If not, then $\mathcal{A}$ outputs $\perp$ to $\mathcal{R}$. Otherwise, $\mathcal{A}$ continues.
2. Adversary $\mathcal{A}$ picks 2^ℓ messages $\mu_{n,1}, \ldots, \mu_{n,2^\ell}$ from the message space $\mathcal{M}$ (e.g., at random or the lexicographically first) and defines the universe $\mathcal{U}_n = (\mu_{n,1}, \ldots, \mu_{n,2^\ell})$.
3. $\mathcal{A}$ picks $i_n \in_R \{1, \ldots, q_s\}, m_n^* \in_R \{\mu_1, \ldots, \mu_{2^\ell}\}$ and $(m_{n,1}, \ldots, m_{n,q_s}) \in_R (\{\mu_1, \ldots, \mu_{2^\ell}\} \backslash \{m_n^*\})^{q_s}$. This defines the following two sequences of integers:

$$\mathcal{M}_n^{\text{rw}} = (m_{n,1}, \ldots, m_{n,q_s}), \quad \mathcal{M}_n^{\text{first}} = (m_{n,1}, \ldots, m_{n,i_n-1}, m_n^*).$$

4. Adversary $\mathcal{A}$ then queries the signing oracle on the messages $\mathcal{M}_n^{\text{first}}$ (i.e., $(m_{n,1}, \ldots, m_{n,i_n-1}, m_n^*)$) and receives the response $\mathcal{S}_n^{\text{first}}$ which are the signatures, with the last σ^* being the signature on the message indexed by m_n^*, if the reduction does not abort. We stress that these are all valid signatures, which we can check using the public verification key. If any of the signatures is not valid, and the reduction does not abort, then we would abort.
5. Adversary $\mathcal{R}$ is then rewound back to before it answered the signature queries. Now $\mathcal{A}$ queries the signing oracle on the messages indexed by $\mathcal{M}_n^{\text{rw}}$ (i.e., $(m_1, \ldots, m_{q_s})$) and receives $\mathcal{S}^{\text{rw}}$ which contains the signatures of the messages indexed by $\mathcal{M}_n^{\text{rw}}$, if the reduction does not abort. We stress that these are all valid signatures, which we can check using the public verification key. If any of the signatures is not valid, and the reduction does not abort, then we would abort.
6. Adversary $\mathcal{A}$ then tosses a biased coin τ_n with probability $\varepsilon_{\mathcal{F}}$ of returning 1, and if $\tau_n = 0$, then $\mathcal{I}$ sends $\perp$ to $\mathcal{R}$. If $\tau = 1$, then $\mathcal{I}$ submits (m_n^*, σ_n^*) as a forgery. Note that this is done in time $t_{\mathcal{F}}$ in order to correctly simulate a forger.
7. Because the reduction was rewound, this constitutes a valid forgery, as m_n^* was not queried to the signature oracle and σ_n^* is indeed a valid signature on m_n^*.

This process is repeated for each round, with $n = 1$ to r. We note that in each round, we need to draw a new set of messages for the universe $\mathcal{U}_n$, as the message space may have changed, due to the public key being different. After at most r rounds, the reduction $\mathcal{R}$ outputs a solution $\widehat{S}$ with probability $\varepsilon_{\mathcal{R}}$, which $\mathcal{A}$ will use as its solution to I.

We now analyze the success probability of $\mathcal{A}$ in breaking the search problem Π. For the nth round, we define $\mathcal{Q}_n$ as the set of sequences of indices such that the corresponding signature queries are correctly answered by $\mathcal{R}$ in the nth round. We can see that if a sequence is correctly answered by $\mathcal{R}$ in the nth round, then the same sequence without the last query is also correctly answered by $\mathcal{R}$ in the nth round. That is to say, for any sequence $(m_{n,1}, \ldots, m_{n,j}) \in \mathcal{Q}_n$, we have $(m_{n,1}, \ldots, m_{n,j-1}) \in \mathcal{Q}_n$.

Consider now a real forger $\mathcal{F}_{\text{real}}$, which is run r times and in the nth round it receives with public keys pk_n and makes signature queries corresponding to m_n, receives $\mathcal{S}_n^{\text{rw}}$ as an answer, and then submits $(m_{m_n^*}, \sigma_n^*)$ as forgery, with probability $\varepsilon_{\mathcal{F}_{\text{real}}}$. By definition, after at most the rth round, the reduction will output a candidate solution S with probability $\varepsilon_{\mathcal{R}}$. After each nth rewind, the reduction $\mathcal{R}$ sees the same transcript, in the nth round, when interacting with $\mathcal{F}_{\text{sim}}$ as it does when it interacts with $\mathcal{F}_{\text{real}}$, unless $\mathcal{M}_n^{\text{first}} \notin \mathcal{Q}_n$ and $\mathcal{M}_n^{\text{rw}} \in \mathcal{Q}_n$. In this case, the real forger $\mathcal{F}_{\text{real}}$ will output a forgery with probability $\varepsilon_{\mathcal{F}_{\text{real}}}$; however, the forger simulated by $\mathcal{A}$ will be unable to output a forgery. This relies on the fact that pk_n defines a unique signature scheme, and hence, the forgery output by the forger $\mathcal{F}_{\text{real}}$ would be the same as the simulated one.

We denote the execution of the reduction with the real forger as $\mathcal{R}^{\mathcal{F}_{\text{real}}}$ and the execution of the reduction with the forger simulated by $\mathcal{A}$ as $\mathcal{R}^{\mathcal{F}_{\text{sim}}}$. From above, we get:

$$\begin{aligned}
&\Pr[\mathcal{R}^{\mathcal{F}_{\text{real}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] - \Pr[\mathcal{R}^{\mathcal{F}_{\text{sim}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] \\
&\quad \leq \sum_{n=1}^{r} \varepsilon_{\mathcal{F}} \cdot \Pr[\mathcal{M}_n^{\text{rw}} \in \mathcal{Q}_n \wedge \mathcal{M}_n^{\text{first}} \notin \mathcal{Q}_n]
\end{aligned}$$

By Lemma 9 we have

$$\begin{aligned}
&\Pr[\mathcal{R}^{\mathcal{F}_{\text{real}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] - \Pr[\mathcal{R}^{\mathcal{F}_{\text{sim}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] \\
&\quad \leq \sum_{n=1}^{r} \varepsilon_{\mathcal{F}} \cdot \Pr[\mathcal{M}_n^{\text{rw}} \in \mathcal{Q}_n \wedge \mathcal{M}_n^{\text{first}} \notin \mathcal{Q}_n]
\end{aligned}$$

Overall, we obtain for the success probability $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$:

$$\begin{aligned}
\varepsilon_{\mathcal{A}} &\geq \Pr[\mathcal{R}^{\mathcal{F}_{\text{sim}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] \\
&\geq \Pr[\mathcal{R}^{\mathcal{F}_{\text{real}}}(I) \rightarrow S : \mathsf{Verify}(I, S, w) = 1] - \varepsilon_{\mathcal{F}} \cdot \frac{r \cdot \exp(-1)}{q_s} \left(1 - \frac{q_s}{2^{\ell}}\right)^{-1} \\
&= \varepsilon_{\mathcal{R}} - \varepsilon_{\mathcal{F}} \cdot \frac{r \cdot \exp(-1)}{q_s} \left(1 - \frac{q_s}{2^{\ell}}\right)^{-1}.
\end{aligned}$$

Now we consider the case where the reduction can rewind the forger to some state St. This is equivalent to restarting the forger with the same random coins and giving it the same input as before until it reaches the required state St. If the reduction rewinds the forger in the $(n + 1)$th round and sends the same public key to the forger, the forger will make the same signature queries and submit the same forgery as in the previous round. This can be simulated by $\mathcal{A}$ by setting $\mathcal{U}_{n+1} = \mathcal{U}_n, m_{n+1}^* = m_n^*, \mathcal{M}_{n+1}^{\text{rw}} = \mathcal{M}_n^{\text{rw}}, \mathcal{M}_{n+1}^{\text{first}} = \mathcal{M}_n^{\text{first}}$ and proceeding as before. However, in the case where the reduction sends a new public key, then the forger will make different queries and submit a different forgery. Adversary $\mathcal{A}$ can simulate this by picking new messages and new values for $m_{n+1}^*, \mathcal{M}_{n+1}^{\text{rw}}, \mathcal{M}_{n+1}^{\text{first}}$, as it normally does, and proceeding as before. We see that in both cases the transcript the reduction sees in the $(n + 1)$th round when interacting with $\mathcal{A}$ is the same as when interacting with a real forger $\mathcal{F}_{\text{real}}$, except when

$\mathcal{M}_{n+1}^{\mathrm{rw}} \in \mathcal{Q}_{n+1} \wedge \mathcal{M}_{n+1}^{\mathrm{first}} \notin \mathcal{Q}_{n+1}$. By a similar argument as above, we get the same bound.

We see that the adversary runs each round of the reduction twice and simulates the forger twice, thus giving us our time bound. □

Remark 2. We need that the forgers are run sequentially, otherwise the proof might not go through. When the reduction is allowed to run the forgers in parallel, it may interleave the executions of the forgers and make the replies to one dependant on the queries of the others. If this is the case, then real forgers would not have a problem producing a forgery. However, the simulation of the forger as described above may not be able to do so. Due to the fact that the reduction may change its responses based on the queries, the simulated forgers may never be able to get a forgery and then submit it.

As an example, consider the following execution of the adversary described above, with only 2 parallel forgers. The adversary sends the instance I to the reduction. The reduction in reply sends a public key pk_1 for the first forger $\mathcal{F}_1$. The first forger defines the universe $\mathcal{U}_1$ and sequences $\mathcal{M}_1^{\mathrm{rw}}, \mathcal{M}_1^{\mathrm{first}}$ as defined above. It then proceeds to make signing queries on messages indexed $\mathcal{M}_1^{\mathrm{first}}$. Based on these queries, $\mathcal{R}$ computes a public key pk_2 for a second forger $\mathcal{F}_2$. The second forger begins by defining its universe $\mathcal{U}_2$ and sequences $\mathcal{M}_2^{\mathrm{rw}}, \mathcal{M}_2^{\mathrm{first}}$. Forger $\mathcal{F}_2$ then proceeds to make signing queries on the messages indexed by $\mathcal{M}_2^{\mathrm{first}}$. Based on this the reduction $\mathcal{R}$ computes the responses $\mathcal{S}_1^{\mathrm{first}}$ and sends it to forger $\mathcal{F}_1$. Having received this, adversary $\mathcal{A}$ rewinds $\mathcal{R}$ to before it answered the signature queries. At this point forger $\mathcal{F}_1$ will make signature queries to the messages indexed by $\mathcal{M}_1^{\mathrm{rw}}$. Using these new queries, the reduction $\mathcal{R}$ will compute a new public key $\widehat{pk}_2$ and use it to initialize forger $\mathcal{F}_2$. Due to the fact that $\widehat{pk}_2 \neq pk_2$, forger $\mathcal{F}_2$ will have to define a new universe $\widehat{\mathcal{U}_2}$ and new sequences $\widehat{\mathcal{M}_2^{\mathrm{first}}}, \widehat{\mathcal{M}_2^{\mathrm{rw}}}$. At this point forger $\mathcal{F}_2$ will make signature queries on the messages indexed by $\widehat{\mathcal{M}_2^{\mathrm{first}}}$. Based on these queries, the reduction $\mathcal{R}$ will compute the responses to forger $\mathcal{F}_1$. At this point it is not clear how to proceed, as the internal state of the reduction has changed. In particular the set $\mathcal{Q}_1'$ may now exclude $\mathcal{M}_1^{\mathrm{rw}}$, which would mean that we can no longer apply Lemma 9. Extending Theorem 4 to parallel forgers remains an open problem.

6. The Probabilistic Signature Scheme

6.1. *The Scheme*

Let $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ be a trapdoor permutation. For simplicity we assume that $\mathsf{Dom}_{pub} = \{0,1\}^k$, for all pub output by Gen. (If that is not the case and we have, e.g., $\mathsf{Dom}_{pub} = \mathbb{Z}_N^*$, the scheme can be adapted accordingly [4].) We now recall the Probablistic Signature Scheme (PSS) [4] PSS is parametrized not only by the security parameter, but also by two additional integers k_0 and k_1. The first parameter k_0 defines the size of randomness in bits and the second defines the domains of the two hash functions. PSS uses two hash functions $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{k_1}$ and $\mathsf{F} : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1}$. We also require two additional functions F_1 and F_2. $\mathsf{F}_1(\omega)$ returns the first k_0 bits of the

```
procedure KeyGen                                                          TDP-PSS
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$
Pick hash functions $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{k_1}$, $\mathsf{F} : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1}$
return $(pk = (\mathsf{H}, \mathsf{F}, pub), sk = td)$

procedure Sign$(sk, m)$
$r \in_R \{0,1\}^{k_0}$
$\omega \leftarrow \mathsf{H}(m||r)$
$r^* \leftarrow \mathsf{F}_1(\omega) \oplus r$
$y = \omega \| r^* \| \mathsf{F}_2(\omega)$
return $\sigma = \mathsf{Invert}(td, y)$                                 // $\sigma = f^{-1}(y)$

procedure Verify$(pk, m, \sigma)$
$y = \mathsf{Eval}(pub, \sigma)$
parse $y$ as $\omega || r^* || \gamma$
$r = r^* \oplus \mathsf{F}_1(\omega)$
if $\mathsf{H}(m||r) = \omega \wedge \mathsf{F}_2(\omega) = \gamma$
    return 1
else
    return 0
```

Fig. 10. The trapdoor permutation probabilistic signature scheme TDP-PSS.

output of $\mathsf{F}(\omega)$ and $\mathsf{F}_2(\omega)$ returns the remaining $(k - k_1) - k_0$ bits of $\mathsf{F}(\omega)$. The scheme TDP-PSS$[k_0, k_1]$ is defined in Fig. 10.

6.2. *Classical Security Results of TDP-PSS*

The original reduction by Bellare and Rogaway from one-wayness of TDP with a security loss of q_h [4] was later improved by Coron to a factor of q_s [14] for the case of the RSA trapdoor permutation.

Theorem 5. (Coron [14]) *Assume the trapdoor permutation* RSA *is* (t', ε')*-hard to invert. Then for any* (q_h, q_s), RSA-PSS$[k_0, k_1]$ *is* $(q_h, q_s, t, \varepsilon)$-UF-CMA *secure in the Random Oracle Model, where*

$$\varepsilon = \varepsilon' \left(1 + 6 \cdot q_s \cdot 2^{-k_0}\right) + 2 \cdot (q_h + q_s)^2 \cdot 2^{-k_1}$$
$$t = t' - (q_h + q_s + 1) \cdot k_1 \cdot \mathcal{O}(k^3).$$

6.3. *A Tight Security Proof for TDP-PSS*

We now present a tight reduction which reduces breaking TDP-PSS to the lossiness of the underlying trapdoor permutation TDP.

```
procedure Initialize                                   Games $\mathsf{G}_0 - \mathsf{G}_4$
$(pub, td) \leftarrow_{\$} \mathsf{Gen}(1^k)$                  $//\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_4$
$(pub, \perp) \leftarrow_{\$} \mathsf{LossyGen}(1^k)$          $//\mathsf{G}_2, \mathsf{G}_3$
Return $pk = pub$

procedure H$(m, r)$                                    Game $\mathsf{G}_0$
if $\mathcal{H}[m, r]$ is defined then
    fetch $\omega_{m,r} = \mathcal{H}[m, r]$
else
    $\omega_{(m,r)} \in_R \{0,1\}^{k_1}$
    $\mathcal{H}[m, r] := \omega_{(m,r)}$
end if
return $\omega_{(m,r)}$

procedure Sign$(m)$                                    Game $\mathsf{G}_0$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
$r \in_R \{0,1\}^{k_0}$
$\omega =$H$(m, r)$
$r^* = \mathsf{F}_1(\omega) \oplus r$
$y = \omega \| r^* \| \mathsf{F}_2(\omega)$
return $\sigma_m = \mathsf{Invert}(td, y)$

Procedure Finalize$(m^*, \sigma^*)$                   Game $\mathsf{G}_0 - \mathsf{G}_4$
$y = \mathsf{Eval}(pub, \sigma^*)$; parse $y = \omega || r^* || \gamma$
$r = r^* \oplus \mathsf{G}_1(\omega)$; Call H$(m, r)$
Fetch $\mathcal{H}[(m^*, r] = (\omega_{(m,r)}, \sigma_{(m,r)})$                    $//\mathsf{G}_3, \mathsf{G}_4$
if $\sigma_{(m^*,r)} = \sigma^*$ then BAD = true then return 0 $//\mathsf{G}_3, \mathsf{G}_4$
if $(\mathsf{H}(m^*, r) = \omega) \wedge (\mathsf{G}_2(\omega) = \gamma) \wedge (m^* \notin \mathcal{M})$
    then return 1
else
    return 0
```

```
procedure F$(\omega)$                                  Game $\mathsf{G}_0 - \mathsf{G}_4$
if $\mathcal{W}[\omega]$ is defined then
    fetch $\alpha = \mathcal{W}[\omega]$
else
    $\alpha \in_R \{0,1\}^{k-k_1}$
    $\mathcal{W}[\omega] := \alpha$
end if
return $\alpha$

procedure H$(m, r)$                                    Games $\mathsf{G}_1 - \mathsf{G}_4$
if $\mathcal{H}[m, r]$ is defined then
    fetch $(\omega_{(m,r)}, \sigma_{(m,r)}) = \mathcal{H}[m, r]$
else
    $\sigma_{(m,r)} \in_R \mathsf{Dom}_{pub}$
    $y_{(m,r)} = \mathsf{Eval}(pub, \sigma_{(m,r)})$
    Parse $y_{(m,r)} = \omega_{(m,r)} || r^*_{(m,r)} || \gamma_{(m,r)}$
    $\alpha_{m,r} = (r^*_{(m,r)} \oplus r) || \gamma_{m,r}$
    if $\mathcal{W}[\omega_{(m,r)}]$ is defined then
        if $\mathcal{W}[\omega_{(m,r)}] \neq \alpha_{m,r}$ then return $\perp$
        endif
    $\mathcal{W}[\omega_{(m,r)}] := \alpha_{m,r}$
    $\mathcal{H}[m, r] := (\omega_{(m,r)}, \sigma_{(m,r)})$
    endif
end if
return $\omega_{(m,r)}$

procedure Sign$(m)$                                    Games $\mathsf{G}_1 - \mathsf{G}_4$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
$r \in_R \{0,1\}^{k_0}$
Call $\mathsf{H}(m, r)$
Fetch $\mathcal{H}[m, r] = (\omega_{m,r}, \sigma_{m,r})$, return $\sigma_{m,r}$
```

Fig. 11. Games for the proof of Theorem 3.

Theorem 6. *Assume* $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *is a regular* (l, t', ε')*-lossy trapdoor permutation for* $l \geq 2$. *Then, for any* (q_h, q_s, k_0, k_1), $\mathsf{TDP\text{-}PSS}[k_0, k_1]$ *is* $(t, \varepsilon, q_h, q_s)$*-* $\mathsf{UF\text{-}CMA}$ *secure in the Random Oracle Model, where*

$$\varepsilon = \left(\frac{2l-1}{l-1}\right) \cdot \varepsilon' + \frac{(q_h + q_s + 1)^2}{2^{k_1}}$$
$$t = t' - q_h \cdot T_{\mathsf{TDP}},$$

and T_{TDP} *is the time to evaluate* TDP.

Remark 3. We stress that the security reduction of Theorem 6 is independent of the size of the randomness k_0. In particular, this allows to set $k_0 = 0$.

Proof. We prove our theorem using a series of games. The description of these games is found in Fig. 11. Let $\mathcal{A}$ be an adversary that runs in time t against $\mathsf{TDP\text{-}PSS}$ executed in the $\mathsf{UF\text{-}CMA}$ experiment described in G_0 in Fig. 11 with $\varepsilon = \Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{UF\text{-}CMA}^{\mathcal{A}} \Rightarrow 1]$.

Now consider Game G_1 from Fig. 9.

Lemma 10. $\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] \geq \Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \frac{(q_h+q_s+1)^2}{2^{k_1}}$.

Proof. In G_0, we modeled the hash functions as a random oracle. In G_1, we modify the H oracle and the signing oracle. On any (m, r), the H oracle now works by evaluating the permutation on a random element $\sigma_{(m,r)} \in \mathsf{Dom}_{pub}$. The result is then parsed as $\omega_{(m,r)}||r^*_{(m,r)}||\gamma_{(m,r)}$ and we check whether $\omega_{(m,r)}$ has been previously queried to the F oracle. If it has, then we abort, otherwise we store the randomness and the values $\omega_{(m,r)}$ and $\sigma_{(m,r)}$. Then, we compute the output of $\mathsf{F}(\omega_{(m,r)})$ and store it. The signing oracle is then modified to pick a random value r, for which we have already have computed $\mathsf{H}(m||r)$, and return the element $\sigma_{(m,r)}$. Note that signing no longer requires the trapdoor td. It can be seen that all our signatures will verify due to the manner in which we compute the responses to queries to H and F. Thus, our simulation of the signatures is correct. Since TDP is a permutation, the distribution of our H-Oracle queries in G_1 is the same as in G_0, except when the reduction aborts, due to a collision in the F-Oracle. The probability of a collision is at most $\frac{(q_s+q_h+1)^2}{2^{k_1}}$, since the adversary makes at most $q_h + q_s + 1$ queries, implicit or explicit, to the F-oracle, giving at most $(q_h + q_s + 1)^2$ possible collisions from a total of 2^{k_1} possible choices. Furthermore, we do not abort if collision matches the value already set, which happens with a probability $1 - \frac{1}{2^{k-k_1}}$. Hence, the total probability that we about is $\left(1 - \frac{1}{2^{k-k_1}}\right)\left(\frac{(q_h+q_s+1)^2}{2^{k_1}}\right)$. Thus, we have $\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] \geq \Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{(q_h+q_s+1)^2}{2^{k_1}}$. □

Lemma 11. *There exists a distinguisher $\mathcal{D}_1$ against the lossiness of* TDP*, which runs in time $t = t_{\mathcal{A}} + (q_h+q_s)\cdot T_{\mathsf{TDP}}$ and that* $\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - |\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]| = \boldsymbol{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1)$.

Proof. From G_1 to G_2, we change the key generation from a normal permutation to a lossy permutation; however, the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_1$ against the lossiness of TDP, using these games. The distinguisher will run $\mathcal{A}$ and simulates the oracles $\mathsf{Sign}(\cdot)$, $\mathsf{H}(\cdot,\cdot)$, $\mathsf{F}(\cdot)$ as described in games G_1&G_2, for which it requires time $(q_h + q_s) \cdot T_{\mathsf{TDP}}$. Note that $\mathcal{D}_1$ does not require the trapdoor td to simulate the oracles. After $\mathcal{A}$ calls $\mathsf{Finalize}$, $\mathcal{D}_1$ returns the output of $\mathsf{Finalize}$. Thus, we can see that $\Pr[\mathrm{L}_0^{\mathcal{D}_1} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]$. Similarly, we have $\Pr[\mathrm{L}_1^{\mathcal{D}_1} \Rightarrow 1] = \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$. Hence, we have $|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]| = |\Pr[\mathrm{L}_1^{\mathcal{D}_1} \Rightarrow 1] - \Pr[\mathrm{L}_0^{\mathcal{D}_1} \Rightarrow 1]| = \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1)$. □

Lemma 12. $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \left(\frac{l-1}{l}\right)\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$.

Proof. In G_3, we introduce a new rule, which sets BAD to true if the forgery σ^* provided by $\mathcal{A}$ is the same as the simulated signature $\sigma_{m^*,r}$ for the target message m^* and randomness r. If this is the case, the adversary loses the game, i.e., G_3 outputs 0. $\sigma_{m^*,r}$ is independent of $\mathcal{A}$'s view and is uniformly distributed in the set of pre-images of $y_{m^*,r}$. Due to the l regular lossiness of TDP, the probability of a collision is equal to exactly $1/l$. Thus, we see that the BAD rule reduces the probability of the adversary winning the game by $1/l$, hence $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = (1 - \frac{1}{l})\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] = \left(\frac{l-1}{l}\right)\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]$. □

Lemma 13. *There exists a distinguisher $\mathcal{D}_2$ against the lossiness of* TDP*, which runs in time* $t = t_{\mathcal{A}} + (q_h + q_s) \cdot T_{\mathsf{TDP}}$ *and that* $|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]| = \boldsymbol{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2)$.

Proof. From G_3 to G_4, we change the key generation from a lossy permutation to a normal permutation; however, the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_2$ against the lossiness of TDP, using these games. The distinguisher will act as the challenger to $\mathcal{A}$. It will simulate the oracles as described in games G_3&G_4, for which it requires time $(q_h + q_s) \cdot T_{\mathsf{TDP}}$. After $\mathcal{A}$ calls $\mathsf{Finalize}$, $\mathcal{D}_2$ returns the output of $\mathsf{Finalize}$. We can see that $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{L}_0^{\mathcal{D}_2} \Rightarrow 1]$. Similarly, we have $\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{L}_1^{\mathcal{D}_2} \Rightarrow 1]$. Hence, we have $|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]| = |\Pr[\mathrm{L}_1^{\mathcal{D}_2} \Rightarrow 1] - \Pr[\mathrm{L}_0^{\mathcal{D}_2} \Rightarrow 1]| = \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2)$. □

Lemma 14. $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = 0$.

Proof. In G_4, we again use the original KeyGen such that $\mathsf{Eval}(pub, \cdot)$ defines a permutation. This means that our signing function is now a permutation, and thus, any forgery implies a collision. Therefore, whenever the adversary is able to make a forgery, the game outputs 0 due to the BAD rule. Whenever they are unable to make a forgery, the game outputs 0. Thus, we can see that in all cases, the game will output 0, hence $\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = 0$. □

We combine Lemmas 10 to 14 to get:

$$\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_1) + \left(\frac{l}{l-1}\right)\mathbf{Adv}^{\mathsf{L}}_{\mathsf{TDP}}(\mathcal{D}_2) + \frac{(q_s + q_h)^2}{2^{k_1}}.$$

where l is the lossiness of TDP. Because the distinguishers run in the same time, we know that both distinguishers can have at most an advantage of ε', giving us:

$$\varepsilon \leq \frac{2l-1}{l-1} \cdot \varepsilon' + \frac{(q_h + q_s + 1)^2}{2^{k_1}}.$$

This completes the proof. □

6.4. *PSS with Message Recovery*

We now recall the PSS with message recovery (PSS-R) scheme, which a digital signature scheme with message recovery based on PSS.

Definition 8. A digital signature scheme with message recovery is a triple of probabilistic algorithms $\mathsf{SIG\text{-}R} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Recover})$, where:

1. KeyGen takes as an input the unary representation of our security parameter (1^k) and outputs a signing key sk and verification key pk.
2. Sign takes as input a signing key sk, message m and outputs an "enhanced signature" σ.

```
procedure KeyGen                                                    TDP-PSS-R
$(pub, td) \leftarrow_\$ \mathsf{Gen}(1^k)$
Pick hash functions $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$, $\mathsf{F} : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_1}$
return $(pk = (\mathsf{H}, \mathsf{F}, pub), sk = td)$

procedure Sign$(sk, m)$
$r \in_R \{0,1\}^{k_0}$
$\omega \leftarrow \mathsf{H}(m||r)$
$r^* \leftarrow \mathsf{F}_1(\omega) \oplus r$
$m^* \leftarrow \mathsf{F}_2(\omega) \oplus m$
$y = \omega||r^*||m^*$
return $\sigma = \mathsf{Invert}(td, y)$                                  // $\sigma = f^{-1}(y)$

procedure Recover$(pk, \sigma)$
$y = \mathsf{Eval}(pub, \sigma)$
parse $y$ as $\omega||r^*||m^*$
$m = m^* \oplus \mathsf{F}_2(\omega)$
$r = r^* \oplus \mathsf{F}_1(\omega)$
if $\mathsf{H}(m||r) = \omega$
    return $m$
else
    return $\perp$
```

Fig. 12. The trapdoor probabilistic signature scheme with recovery TDP-PSS-R.

3. Recover is a deterministic algorithm, which on input of a public key and an "enhanced signature" σ outputs either the corresponding message m (accept) or the special symbol $\perp$ (reject).

We say that SIG-R is correct if for all public key and secret key pairs generated by KeyGen, we have:

$$\Pr[\mathsf{Recover}(pk, \mathsf{Sign}(sk, m)) = m] = 1.$$

We describe TDP-PSS-R in Fig. 12 using the same notation as in Sect. 6. The overhead of signature scheme with recover is defined as the difference in size between the message and the signature. With PSS-R, we can sign and recover messages of size at most $n = k - k_0 - k_1$ bits long with a k bit signature. Thus, we have a total overhead of $k_1 + k_0$ bits.

Theorem 7. *Assume* $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *is a regular* (l, t', ε')*-lossy trapdoor permutation for* $l \geq 2$*. Then, for any* (q_h, q_s, k_0, k_1), $\mathsf{TDP\text{-}PSS\text{-}R}[k_0, k_1]$ *is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\varepsilon = \left(\frac{2l-1}{l-1}\right) \cdot \varepsilon' + \frac{(q_h + q_s + 1)^2}{2^{k_1}}$$
$$t = t' - q_h \cdot T_{\mathsf{TDP}},$$

and T_{TDP} *is the time to evaluate* TDP.

The proof of security for PSS-R is similar to that of PSS, and hence, we do not present it.

We now discuss the choices of parameters k_0, k_1. First off, we notice that the security of TDP-PSS-R does not depend at all on the size of the randomness k_0. Hence, we can simply pick $k_0 = 0$. We note that this does not contradict the result of Coron [14] that states if we use $k_0 = 0$, then our reduction must lose a factor of q_s. This is due to fact that this only holds if TDP-PSS is instantiated with a certified trapdoor permutation.

For the choice of k_1, we need the notion of time-to-success ratios, also sometimes referred to as "work factor", whose definition we now recall.

Definition 9. The work factor WF of an adversary $\mathcal{A}$ with a success probability of $\varepsilon_{\mathcal{A}}$ and running time $t_{\mathcal{A}}$ is given by:

$$\mathsf{WF}(\mathcal{A}) = \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}}.$$

We say that a scheme achieves κ bit security if for all adversaries $\mathcal{A}$, we have $\mathsf{WF}(\mathcal{A}) \geq 2^\kappa$, that is to say, the adversary must perform at least 2^κ expected operations to win the security game. We remark that if we assume any problem is (t, ε)-hard, we have that for all adversaries $\mathcal{A}$ against the problem, we have $\mathsf{WF}(\mathcal{A}) \geq 2^\kappa$.

The condition that $\mathsf{WF} \geq 2^\kappa$ determines our choice of parameters for a given security level κ. Using this and the formulae from Theorem 7, we can compute the value for k_1 such that TDP-PSS has κ-bit security. Using the values from Theorem 7, we get:

$$\frac{t}{\varepsilon} = \frac{t' - q_h \cdot T_{\mathsf{TDP}}}{\left(\frac{2l-1}{l-1}\right) \cdot \varepsilon' + \frac{(q_h+q_s+1)^2}{2^{k_1}}} \geq 2^\kappa$$

If we rearrange this expression, we see that in fact we need:

$$\frac{t'}{\varepsilon'} \frac{l-1}{2l-1} \geq 2^\kappa \quad \text{and} \quad \frac{t \cdot 2^{k_1}}{(q_s + q_h)^2} \geq 2^\kappa.$$

The first requirement is satisfied by our assumption that we have a regular (l, t', ε')-lossy trapdoor permutation, which means $t'/\varepsilon' \geq 2^\kappa$ by assumption. If we look at the second inequality, we see that we require $k_1 \geq \kappa + 2\log(q_h + q_s) - \log t$. However, we also have that $t \geq q_s + q_h$, hence giving us $k_1 \geq \kappa + \log(q_h + q_s)$.

We can perform similar calculations for the proofs of Bellare-Rogaway and Coron, but we omit them here. We compute some concrete figures for the overhead imposed

Table 1. Total overhead using RSA-PSS-R for 80 bit security.

Security proof	k_0	k_1	Total overhead
Bellare-Rogaway [4]	160	160	320
Coron [14]	30	160	190
This work	0	160	160

by each security proof and present them in Table 1. We take $\kappa = 80, q_h = 2^{80}$, $q_s = 2^{30}$.

Acknowledgements

We thank Mihir Bellare, Dennis Hofheinz, and Bertram Poettering for valuable comments on an earlier draft. We also thank the Journal of Cryptography reviewers for their helpful comments. We would also like to thank the reviewer for the Journal of Cryptology for their insightful comments. This work was done while Saqib A. Kakvi was employed at Ruhr-University Bochum. The authors were supported by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation, funded by the German Federal Ministry for Education and Research.

References

[1] M. Abe, J. Groth, M. Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions, in *Advances in Cryptology—ASIACRYPT 2011*, Lecture Notes in Computer Science (Springer, Berlin, 2011), pp. 628–646

[2] M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko, The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *J. Cryptol.* **16**(3), 185–215 (2003)

[3] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in V. Ashby (ed.) *ACM CCS 93: 1st Conference on Computer and Communications Security* (ACM Press, New York, 1993), pp. 62–73

[4] M. Bellare, P. Rogaway, The exact security of digital signatures: how to sign with RSA and Rabin, in U.M. Maurer (ed) *Advances in Cryptology—EUROCRYPT'96*. Lecture Notes in Computer Science, vol. 1070 (Springer, Berlin, 1996), pp. 399–416

[5] M. Bellare, P. Rogaway, The security of triple encryption and a framework for code-based game-playing proofs, in S. Vaudenay (ed) *Advances in Cryptology—EUROCRYPT 2006*. Lecture Notes in Computer Science, vol. 4004 (Springer, Berlin, 2006), pp. 409–426

[6] M. Bellare, M. Yung, Certifying permutations: noninteractive zero-knowledge based on any trapdoor permutation. *J. Cryptol.* **9**(3), 149–166 (1996)

[7] D.J. Bernstein, Proving tight security for Rabin–Williams signatures. In N.P. Smart (ed) *Advances in Cryptology—EUROCRYPT 2008*. Lecture Notes in Computer Science, vol. 4965 (Springer, Berlin, 2008), pp. 70–87

[8] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in C. Boyd (ed) *Advances in Cryptology—ASIACRYPT 2001*. Lecture Notes in Computer Science, vol. 2248 (Springer, Berlin, 2001), pp. 514–532

[9] C. Cachin. Efficient private bidding and auctions with an oblivious third party, in *ACM CCS 99: 6th Conference on Computer and Communications Security* (ACM Press, New York, 1999), pp. 120–127

[10] C. Cachin, S. Micali, M. Stadler, Computationally private information retrieval with polylogarithmic communication, in J. Stern (ed) *Advances in Cryptology—EUROCRYPT'99*. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 402–414

[11] D. Coppersmith, Finding a small root of a univariate modular equation, in U.M. Maurer (ed) Advances in Cryptology—EUROCRYPT'96. *Lecture Notes in Computer Science*, vol. 1070 (Springer, Berlin, 1996), pp. 155–165

[12] J.-S. Coron, On the exact security of full domain hash, in M. Bellare (ed) *Advances in Cryptology—CRYPTO 2000*. Lecture Notes in Computer Science, vol. 1880 (Springer, BErlin, 2000), pp. 229–235

[13] J.-S. Coron, Optimal security proofs for PSS and other signature schemes. Cryptology ePrint Archive, Report 2001/062, 2001. http://eprint.iacr.org/2001/062.

[14] J.-S. Coron, Optimal security proofs for PSS and other signature schemes, in L.R. Knudsen (ed) *Advances in Cryptology—EUROCRYPT 2002*. *Lecture Notes in Computer Science*, vol. 2332 (Springer, Berlin, 2002), pp. 272–287

[15] C. Gentry, P.D. Mackenzie, Z. Ramzan, Password authenticated key exchange using hidden smooth subgroups, in V. Atluri, C. Meadows, A. Juels (eds) *ACM CCS Conference on Computer and Communications Security* (ACM Press, New York, 2005), pp. 299–309

[16] C. Gentry, C. Peikert, V. Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, in R.E. Ladner, C. Dwork (eds) *40th Annual ACM Symposium on Theory of Computing* (ACM Press, New York 2008), pp. 197–206

[17] C. Gentry, Z. Ramzan, Single-database private information retrieval with constant communication rate, in L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung, (eds) *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*.Lecture Notes in Computer Science, vol. 3580 (Springer, Berlin, 2005), pp. 803–815

[18] E.-J. Goh, S. Jarecki, J. Katz, N. Wang, Efficient signature schemes with tight reductions to the Diffie–Hellman problems. *J. Cryptol.* **20**(4), 493–514 (2007)

[19] B. Hemenway, R. Ostrovsky, Public-key locally-decodable codes, in D. Wagner (ed) *Advances in Cryptology—CRYPTO 2008*. *Lecture Notes in Computer Science*, vol. 5157 (Springer, Berlin, 2008), pp. 126–143

[20] IEEE P1363a Committee. IEEE P1363a / D9 — standard specifications for public key cryptography: Additional techniques. http://grouper.ieee.org/groups/1363/index.html/, June 2001. Draft Version 9.

[21] S.A. Kakvi, E. Kiltz, A. May, Certifying RSA, in *Advances in Cryptology—ASIACRYPT 2012*. Lecture Notes in Computer Science (Springer, Berlin, 2012), pp. 404–414

[22] E. Kiltz, A. O'Neill, A. Smith, Instantiability of RSA-OAEP under chosen-plaintext attack, in T. Rabin (ed) *Advances in Cryptology—CRYPTO 2010*. Lecture Notes in Computer Science, vol. 6223 (Springer, Berlin, 2010), pp. 295–313

[23] N. Koblitz, A. Menezes, Another look at security definitions. Cryptology ePrint Archive, Report 2011/343, 2011. http://eprint.iacr.org/2011/343.

[24] N. Koblitz, A.J. Menezes, Another look at "provable security". *J. Cryptol.* **20**(1), 3–37 (2007)

[25] A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham, Sequential aggregate signatures from trapdoor permutations, in C. Cachin, J. Camenisch (eds) *Advances in Cryptology—EUROCRYPT 2004*. Lecture Notes in Computer Science, vol. 3027 (Springer, Berlin, 2004), pp. 74–90

[26] S. Micali, Computationally sound proofs (2000), pp. 1253–1298

[27] P. Paillier, J.L. Villar. Trading one-wayness against chosen-ciphertext security in factoring-based encryption, in X. Lai, K. Chen (eds) *Advances in Cryptology – ASIACRYPT 2006*. Lecture Notes in Computer Science, vol. 4284 (Springer, Berlin, 2006), pp. 252–266

[28] C. Peikert, B. Waters, Lossy trapdoor functions and their applications, in R.E. Ladner, C. Dwork (eds) *40th Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 2008), pp. 187–196

[29] PKCS #1: RSA cryptography standard. RSA Data Security, Inc., Sept. 1998. Version 2.0

[30] C. Schridde, B. Freisleben, On the validity of the phi-hiding assumption in cryptographic protocols, in J. Pieprzyk (ed) *Advances in Cryptology—ASIACRYPT 2008*. Lecture Notes in Computer Science, vol. 5350 (Springer, Berlin, 2008), pp. 344–354

[31] N. Smart, Ecrypt II yearly report on algorithms and keysizes (2009–2010). *Framework* (2010), p. 116