On the Impossibility of Instantiating PSS in the Standard Model

Rishiraj Bhattacharyya[1] and Avradip Mandal[2]

[1] Cryptology Research Group, Applied Statistics Unit, Indian Statistical Institute, Kolkata. rishi_r@isical.ac.in

[2] Université du Luxembourg, Luxembourg. avradip.mandal@uni.lu

Abstract. In this paper we consider the problem of securely instantiating Probabilistic Signature Scheme (PSS) in the standard model. PSS, proposed by Bellare and Rogaway [3] is a widely deployed randomized signature scheme, provably secure (*unforgeable under adaptively chosen message attacks*) in Random Oracle Model.

Our main result is a black-box impossibility result showing that one can not prove unforgeability of PSS against chosen message attacks using blackbox techniques even assuming existence of *ideal trapdoor permutations* (a strong abstraction of trapdoor permutations which inherits all security properties of a random permutation, introduced by Kiltz and Pietrzak in Eurocrypt 2009) or the *lossy trapdoor permutations* [19]. Moreover, we show *onewayness*, the most common security property of a trapdoor permutation does not suffice to prove even the weakest security criteria, namely *unforgeability under zero message attack*. Our negative results can easily be extended to any randomized signature scheme where one can recover the random string from a valid signature.

Keywords: PSS, Blackbox Reductions, Randomized Signature, Standard Model.

1 Introduction

Probabilistic Signature Scheme (PSS) is one of the most known and widely deployed provably secure randomized signature schemes. It was designed by Bellare and Rogaway [3] as a generic scheme based on a trapdoor permutation (like RSA). In [3], Bellare and Rogaway showed the scheme is secure in Random Oracle (RO) Model [2]. Coron improved the previous security bound in [6]. Recently in [7], PSS is proven secure even against fault attacks exploiting the Chinese Remainder Theorem (CRT) implementation of RSA . However, all the previous security proofs are valid only in RO model, where one assumes the existence of ideal, truly random hash functions. Unfortunately truly random functions do not exist and in practice, the "ideal" functions are instantiated with some efficient hash functions. Hence it is important that the proofs are valid while replacing random oracles by a standard hash functions. Otherwise such proofs merely provide heuristic evidence that breaking the scheme may be hard (or there is no generic attack against the scheme).

A number of papers [5, 8, 13, 17], starting from famous results of Canetti et. al. [5], showed that there are schemes secure in the Random Oracle model, which are uninstantiable under standard model. Naturally, these results raise concerns about the soundness of the schemes proven secure in random oracle model. Particularly for widely deployed scheme like PSS, it is especially important to have an secure instantiation by a standard, efficiently computable hash function so that we do not build our technology in vacuum. In this paper, we ask essentially this particular question about PSS: *Whether it is possible, to securely instantiate PSS based on reasonable assumptions to the underlying trapdoor permutation.*

1.1 Our Results

Our main result is a general negative result to the above question. Roughly, we extend *all* the negative results by Dodis et. al. [8] for Full Domain Hash (FDH) to PSS. Specifically, we show the following

- There is no instantiation of PSS such that, *unforgeability under chosen message attack* can be reduced to any security property of a random permutation using *black-box* reduction techniques. As a random permutation satisfies almost all reasonable security notions, our result covers many of the standard security notions, like inverting trapdoor permutation on a random point (one-way), finding some bits of pre-image of a random point (partial domain one-wayness), finding correlated inputs etc. Our result is perfectly valid even if the hash functions used in PSS can query the trapdoor permutation and digests are arbitrarily related to the responses.
- We also rule out any black box reduction from recently proposed Lossy Trapdoor Permutations [19]. In Crypto 2010, Kiltz et. al. [16] has proven IND-CPA security of OAEP based on Lossy Trapdoor Permutation. Hence it is important to analyze whether positive result could be possible for PSS.
- We also show that even the weakest security criteria , namely unforgeability under *no message attack* cannot be black-box reduced to the one-wayness of the trapdoor permutation if the randomness space in PSS is "super-polynomial" in security parameter.
- All our results can easily be extended to the scenario when the adversary can invert some points of his choice (with some restrictions) for a fixed bounded number of times.

We would like to mention that our results does not completely rule out the possibility of instantiating PSS in standard model. A "whitebox" reduction, using the code of the adversary, may still exist. On the other hand, it may be possible to show a reduction from other cryptographic functions like homomorphic encryption. Still, we believe our result is important from theoretical point of view as it shows PSS requires special property of underlying trapdoor permutation as opposed to "Only randomness of hash is sufficient" notion of random oracle model.

1.2 Overview of our Technique

We use the technique of two oracles due to Hsio and Reyzin [14] for our separation results. We construct two oracles T and G such that T implements a ideal trapdoor permutation and G can be used to forge the PSS scheme. However, G does not help the attacker to break any security property of the ideal trapdoor permutation. Informally, this ensures that a black-box security proof cannot exist as any such proof should be valid against our T and G.

On a very high level our technique can be seen as an extension of the technique of Dodis et. al. [8] to rule out black box reduction of FDH. Separation from a random permutation is achieved in two steps. As the first step, we instantiate T by permutation chosen uniformly at random from the set of exponentially many permutations. Intuitively, G, the main forger oracle, should output a forgery after checking whether the adversary truly has access to a signer by sending polynomially many challenge messages. However the reduction could design the underlying hash function in such a way, so that the digests of the messages either collide with each other (hence reducing the number of points on which inversion is needed) or the digest is the result one of the evaluation queries made to the trapdoor permutation (hence the reduction can get the signature from the corresponding query by evaluating the hash function). For this reason we define G to output the forgery only if the adversary can produce distinct signatures, which were not a query to the trapdoor permutation during the computation of digests, for all the challenge messages.

In the second step we show that a reduction algorithm (which does not have access to inversion oracle) can not produce valid, signature meeting both the conditions with non-negligible probability. Hence to win any hard game, G is of no use to the adversary. However, we construct an efficient adversary with an access to a valid sign oracle (available in an unforgeability game) that can either find a forgery on its own or can construct signatures satisfying all the conditions of G. We stress that the efficient algorithm in [8], which precomputes all the hash values to check for the conditions, does not work efficiently when the signature scheme is randomized. Specifically, when the random strings are of super-logarithmic length, it is no longer possible for a polynomial time algorithm to compute all possible hash values for even a single message. It might very well happen that the computed digests meet the conditions but the digests on which signer generated the signature do not meet the condition. To solve this problem we use an elegant adaptive "evaluate on the fly" technique where we sample polynomially many random strings and check for the conditions. If the conditions are satisfied for the sampled digests, we repeatedly query the signer with fresh random coins for multiple signatures of same message. We show that, with probability exponentially close to 1, one gets either a set of valid signatures maintaining the conditions from the signer or could find a forgery during the sampling stage.

1.3 Previous Results

A rich body of work [10–13, 15, 20] on blackbox separation exists in the literature starting from the seminal work of Impagliazzo and Rudich [15]. Regarding the separation of random oracle from the standard model, the first result was due to Canetti, Goldreich and Halevi [4, 5] who showed an artificial albeit valid signature scheme that can not be securely instantiated by standard hash functions. Many such results [8, 9, 13, 17] were subsequently published. To obtain our separation results we use the two oracle technique of Hsiao and Reyzin[14]. The most relevant results to our paper is the work of Dodis et. al. [8] and of Kiltz and Pietrzak [17]. In [8], Dodis, Oliviera and Pietrzak showed that the popular Full Domain Hash (FDH) signature scheme can not be instantiated (using blackbox technique) in standard model by a ideal trapdoor permutation. Kiltz and Pietrzak [17] established that there is no blackbox reduction of any padding based CCA secure encryption scheme from ideal trapdoor permutations. In [18], Paillier showed impossibility of reduction of many RSA based signatures including PSS from different security assumptions of RSA. However, their result is based on an additional assumption (namely, instance non-malleability) of RSA. In comparison, our result is more generic as we rule out blackbox reduction from any property of random permutation.

1.4 Differences from Dodis et al's Crypto'05 paper [8]

Although our definition of oracles are quite similar to that in [8], difference comes in when finally implementing a forgery. The technique of [8] is not readily applicable for randomized signatures. Specifically in case of PSS the forger cannot force the signer to choose any particular random string. On the other hand, if the randomness space is super-polynomial the forger cannot pre-compute all the possible value of the hashes of any message. As a result the forger, as defined in [8], cannot output a forgery when G aborts. Our contribution is in constructing adaptive forger that can forge PSS with overwhelming probability even when the randomness space is super-polynomial. Moreover, our technique to rule out black-box reduction to one way trapdoor permutation is completely different. Looking ahead, we show that when the randomness space is of super-polynomial size, no Probabilistic Polynomial-Time Turing Machine (PPTM) can use a random signature (over the choice of random string during signing) of any fixed message to invert the one way trapdoor permutation.

2 Preliminaries

2.1 Notations

Throughout the paper, if x is a string, $|x|$ denotes the length of the string. 1^n denotes the string of n many 1s. If S is a set $|S|$ denotes the cardinality of the set. We use $negl(n)$ to denote any function $\gamma : \mathbb{N} \rightarrow [0, 1]$ where for any constant $c > 0$ there exist n_0 such that for all $n > n_0$; $\gamma(n) < 1/n^c$. We call a function

$f(n)$ to be super-polynomial if for any constant $c > 0$, there exists n_0 such that for all $n > n_0$, $f(n) > n^c$.

2.2 Trapdoor Permutations (TDPs)

Definition 1. *A trapdoor permutation family is a triplet of PPTM* (Tdg, F, F^{-1}). *Tdg is probabilistic and on input 1^n outputs a key-pair $(pk, td) \leftarrow_R Tdg(1^n)$. $F(pk, .)$ implements a permutation f_{pk} over $\{0,1\}^n$ and $F^{-1}(td, .)$ implements the corresponding inverse f_{pk}^{-1}.*

The most standard security property of TDP is *one-wayness* which says that it is hard to invert a random element without knowing the trapdoor. Formally, for any PPTM A

$$Pr[(pk, td) \leftarrow_R Tdg(1^n), x \leftarrow_R \{0,1\}^n : A(f_{pk}(x)) = x] \leq negl(n).$$

Many other security notion for Trapdoor Permutations are known. Like [8, 17], we consider a wide class of security properties using the notion of *δ-hard games*.

2.3 Hard Games

A cryptographic game consists of two PPTMs C (Challenger) and A (Prover) who can interact over a shared tape. After the interaction, C finally outputs a bit d. We say, A wins the game if $d = 1$ and denote it, following [8], by $\langle C, A \rangle = 1$.

Definition 2. *A game defined as above is called δ-hard game if for all PPT A (in the security parameter n) the probability of win , when both C and A has oracle access to t uniform random permutations $\pi_1, \pi_2, \cdots, \pi_t$ over $\{0,1\}^n$, is at most negligible more than δ. Formally C is a δ-*hard *game if for all PPTM A*

$$Adv_C(A, n) = Pr[\langle C^{\pi_1, \pi_2, \cdots, \pi_t}, A^{\pi_1, \pi_2, \cdots, \pi_t} \rangle = 1] \leq \delta + negl(n)$$

The hardness of the game C (denoted by $\delta(C)$) is the minimum δ such that C is δ-hard.

For cryptographic games like one-wayness, partial one-wayness, claw-freeness; $\delta = 0$. For the game of pseudo-randomness $\delta = 1/2$. The notion of δ-hard game was considered in [17] as a generalization of hard games considered in [8]. It was pointed out in [17] that the result of [8] can easily be extended to this notion.

2.4 Ideal Trapdoor Permutations

The notion of Ideal Trapdoor permutation was coined in [17]. To remain consistent with literature, we follow the same notion.

Let $TDP = (Tdg, F, F^{-1})$ be a trapdoor permutation. We say that TDP is secure for δ-hard game C if for all PPTM A, $\mathrm{Adv}_C(A, n) - \delta(C)$ is negligible

even when the random permutations in the definition of hard game is replaced by TDP. Formally, TDP is secure iff,

$$Pr[\langle C^{F(pk_1),F(pk_2),\cdots,F(pk_t)}, A^{F(pk_1),F(pk_2),\cdots,F(pk_t)}\rangle = 1] \leq \delta + negl(n),$$

where $(pk_i, td_i) \leftarrow_R Tdg(1^n)$ *for* $i = 1, \cdots t$.

Definition 3. *TDP is said to be an ideal trapdoor permutation if it is secure for any δ-hard game C.*

We stress that, ideal trapdoor permutation does not exist (see [8] for proof). However as we are proving negative result, showing that PSS cannot be reduced to ideal trapdoor permutation (hence to any hard game) makes our result stronger. This implies that PSS cannot be black-box reduced to security notions like collision resistant hashing, pseudo-random functions, IND-CCA secure public key encryption schemes etc.

2.5 Lossy Trapdoor Permutations(LTDPs)

Lossy Trapdoor Functions were introduced by Peikert et. al. in [19]. In this paper we consider a straightforward generalization to permutations. A family of (n, l) Lossy Trapdoor Permutations (LTDPs) is given by a Tuple $(\mathcal{S}, \mathcal{F}, \mathcal{F}')$ of PPTMs. $\mathcal{S}$ is a sampling algorithm which on input 1 invokes $\mathcal{F}$ and on input 0 invokes $\mathcal{F}'$. $\mathcal{F}$ (called "Injective Mode") describes a usual trapdoor permutation; i.e. it outputs (f, f^{-1}) where f is a permutation over $\{0,1\}^n$ and $f^{-}1$ is the corresponding inverse. $\mathcal{F}'$ (called "Lossy Mode") outputs a function f' on $\{0,1\}^n$ with range size at most 2^l. For any distinguisher D, *LTDP-Advantage* is defined as

$$\mathrm{Adv}^{ltdp}_{(\mathcal{F},\mathcal{F}'),D} = \left| Pr[D^f(.) = 1 : (f, f^{-1}) \leftarrow^R \mathcal{F}] - Pr[D^{f'}(.) = 1 : f' \leftarrow^R \mathcal{F}'] \right|.$$

We call $\mathcal{F}$ "lossy" if it is the first component of some lossy LTDP.

3 Signature Schemes

A signature scheme $(\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ is defined as follows:

- The *key generation algorithm* `Gen` is a probabilistic algorithm which given 1^k, outputs a pair of matching public and private keys, (pk, td).

- The *signing algorithm* `Sign` takes the message M to be signed, the public key pk and the private key td, and returns a signature $\sigma = \mathtt{Sign}_{td}(M)$. The signing algorithm may be probabilistic.

- The *verification algorithm* `Verify` takes a message M, a candidate signature σ' and pk. It returns a bit $\mathtt{Verify}_{pk}(M, \sigma')$, equal to one if the signature is accepted, and zero otherwise. We require that if $\sigma \leftarrow \mathtt{Sign}_{td}(M)$, then $\mathtt{Verify}_{pk}(M, \sigma) = 1$.

3.1 Security of a Signature Scheme

In the *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\texttt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

3.2 Probabilistic Signature Scheme(PSS)

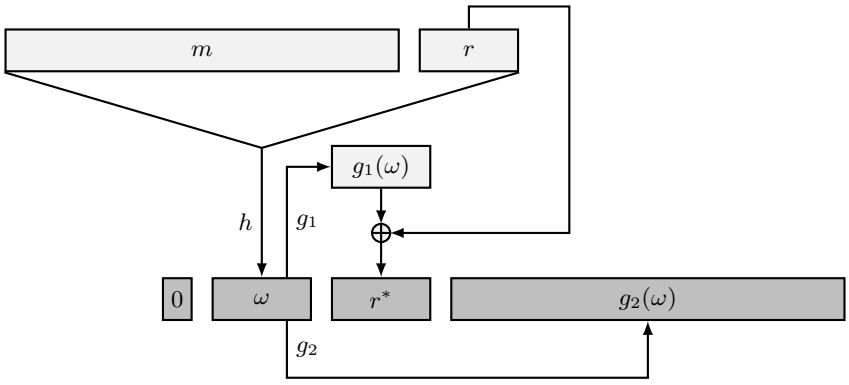

Fig. 1. PSS_H^{TDP}: The components of the image $y = 0\|\omega\|r^*\|g_2(\omega)$ are darkened. The signature of m is $F^{-1}(td, y)$

Let $TDP = (Tdg, F, F^{-1})$ be a trapdoor permutation. PSS uses a triplet $H = (h, g_1, g_2)$ of hash functions such that, $h : \{0,1\}^* \to \{0,1\}^{k_1}$, $g_1 : \{0,1\}^{k_1} \to \{0,1\}^{k_0}$ and $g_2 : \{0,1\}^{k_1} \to \{0,1\}^{k-k_0-k_1-1}$, where k, k_0 and k_1 are parameters.

$\texttt{Gen}(1^k)$

1. Return $(pk, td) = Tdg(1^k)$

$\texttt{Sign}_{td}(m)$

1. $r \leftarrow \{0,1\}^{k_0}$
2. $\omega \leftarrow h(m\|r)$
3. $r^* \leftarrow g_1(\omega) \oplus r$
4. $y \leftarrow 0\|\omega\|r^*\|g_2(\omega)$
5. Return $\sigma = F^{-1}(td, y)$.

$\texttt{Verify}_{pk}(m, \sigma)$

1. Let $y = F(pk, \sigma)$
2. Parse y as $0\|\omega\|r^*\|\gamma$. If the parsing fails return 0.
3. $r \leftarrow r^* \oplus g_1(\omega)$
4. If $h(m\|r) = \omega$ and $g_2(\omega) = \gamma$ return 1.
5. else return 0.

Any PSS signature scheme can be instantiated by specifying the triplet of hash functions $H = (h, g_1, g_2)$ and the trapdoor permutation TDP. PSS_H^{TDP} be the PSS signature scheme instantiated by H and TDP. For any $H = (h, g_1, g_2)$, the PSS transformation described above is defined as

$$PSS_H^{f_{pk}}(m\|r) = 0\|h(m\|r)\|(r \oplus g_1(h(m\|r))\|g_2(h(m\|r)).$$

$PSS_H^{f_{pk}}(m\|r)$ is in fact the darkened area in Figure 1, $y = 0\|\omega\|r^*\|g_2(\omega)$. *Note, h, g_1, g_2 can be oracle circuits with oracle access to f_{pk}.* For the rest of the paper, PSS_H^{TDP} denotes the signature scheme, where as $PSS_H^{f_{pk}}(\cdot)$ is the PSS transformation during `Sign` procedure before applying the trapdoor permutation. From the context these two notations are easily distinguishable.

The following observation is very important to our technique.

Observation 1 *A collision after the PSS transformation implies collision in the random space. In other words,*

$$PSS_H^{f_{pk}}(M_1\|r_1) = PSS_H^{f_{pk}}(M_2\|r_2)$$

implies $r_1 = r_2$.

As both the digests are same, $\omega_1\|r_1^*\|\gamma_1 = \omega_2\|r_2^*\|\gamma_2$; we have $\omega_1 = \omega_2$ and $r_1^* = r_2^*$. This leads to $r_1 = r_2$. So for two distinct random strings r_1 and r_2, the digests of $PSS_H^{f_{pk}}$ and hence the signatures are always different (irrespective of whether the messages are same or not)! As a side note, all our results are valid not only for PSS, but for any randomized signature scheme where the randomness is recoverable and hence the Observation 1 holds true.

4 No Blackbox Reduction from One way Trapdoor Permutations

One-wayness is the most common security property of a trapdoor permutation. All the previous security proofs of PSS in Random Oracle model are based on one wayness of underlying trapdoor permutation (specifically RSA). In this section

we consider the possibility of reducing security of PSS from one-wayness of a trapdoor permutation, but in standard model. We show that when $k_0 = \omega(\log n)$, one cannot prove PSS secure via a blackbox reduction from one way trapdoor permutation even if the forger is never allowed to query the signer.

Recall that, $r_1 \neq r_2$ implies $PSS_H^{f_{pk}}(0\|r_1) \neq PSS_H^{f_{pk}}(0\|r_2)$. So the set $\{PSS_H^{f_{pk}}(0\|r)|r \in \{0,1\}^{k_0}\}$ is of super-polynomial size. Even if G returns one random signature (from a choice of superpolynomially many) of message 0, it is unlikely to be of any use of the adversary intended to invert TDP^T on a uniformly chosen element z.

Following [14], Proposition 1, to rule out blackbox reductions, it is enough to construct two oracles T and G such the following holds:

- There exists an oracle PPTM TDP such that TDP^T implements a trapdoor permutation.
- There exist an oracle PPTM A such that $A^{T,G}$ finds a forgery under chosen message attack for $PSS_H^{TDP^T}$.
- TDP^T is an one-way trapdoor permutation relative to the oracles T and G. That is, TDP^T is an one-way permutation even if the adversary is given oracle access to T and G.

Definition of T For any $n \in \mathbb{N}$, Choose $2^n + 1$ permutations $f_0, f_1, f_2, \cdots, f_{2^n-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^n$. Now the oracle T is defined as follows:

- $T_1(td) \rightarrow g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \rightarrow f_{pk}(y)$ (evaluate)
- $T_3(td, z) \rightarrow f^{-1}_{g(td)}(z)$ (inversion)

Implementing TDP^T We use $T = (T_1, T_2, T_3)$ in the following way to construct (in the functional sense) the trapdoor permutation $TDP^T = (Tdg, F_T, F_T^{-1})$.

- $Tdg(1^n)$ chooses a uniform random $td \leftarrow \{0,1\}^n$ and computes the corresponding public key as $pk = T_1(td)$ and outputs (td, pk).
- $F_T(pk, y)$ returns $T_2(pk, y)$.
- $F_T^{-1}(td, z)$ returns $T_3(td, z)$.

It is easy to check that as TDP^T implements a trapdoor permutation, as $g(td) = pk$.

Description of G The oracle G takes as input $k \in \mathbb{N}$ and $H \in \{0,1\}^*$. G selects an $r \in_R \{0,1\}^{k_0}$ and returns $f_{pk}^{-1}(PSS_H^{f_{pk}}(0\|r))$. Here pk is the public key generated by $Tdg(1^k)$.

As G always outputs a forgery for message 0, we get the following result.

Lemma 1. *There is a PPTM A such that A^G outputs a forgery for PSS signature scheme.*

G does not break security of TDP^T

Next we shall prove that TDP^T is one way, even relative to G. This is not at all obvious as G always provides forgery of the form $f_{pk}^{-1}(PSS_H^{f_{pk}}(.))$ for a H of our choice! But we note that $G(.)$ samples one z' from a set of superpolynomial size and outputs $f_{pk}^{-1}(z)$. Even if the adversary sets $PSS_H^{f_{pk}}(0,r)$ for one r to be the challenge z she received, probability that $f_{pk}(G(.)) = z$ is negligible. On the other hand if $f_{pk}(G(.)) \neq z$, then knowledge of inverse of some other point does not help the adversary to find $f_{pk}^{-1}(z)$ with significant probability for a pseudorandom f_{pk}. Following the above discussion we have Lemma 2,

Lemma 2. *A random permutation $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$ is one way even if adversary is allowed to make one inverse query on any input except the challenge.*

Proof. Suppose, for contradiction, the lemma is not true. Then there exist a PPTM $B = (B_1, B_2)$ such that

$$Pr[B_1^{\pi}(z) = z'; B_2^{\pi}(z, z', \pi^{-1}(z')) = x; z \neq z' : \pi(x) = z] \geq 1/n^c$$

for some constant $c > 0$. Now We shall construct a PPTM B' that , using B, can invert a random permutation $\pi' : T \rightarrow T$ where $|T| = 2^n - 1$. First we note that we can view T as the set $\{0,1\}^n \setminus 1^n$. B' keeps a list L with all the query responses. Without loss of generality, we assume B does not repeat a query and if B outputs some x it must have queried it.

Suppose B_1 and B_2 makes n^{c_1} and n^{c_2} queries respectively (c_1, c_2 are positive constant). B' works as following: on receiving the challenge z, check whether $z = 1^n$. If yes, abort; otherwise simulate $B_1(z)$. Clearly the probability that a randomly chosen z is equal to 1^n is $1/2^k$; hence negligible. When B_1 makes an oracle query x, check whether $x = 1^n$. If yes, select one element y_1 uniformly at random from $\{0,1\}^n \setminus L_y$. If $y_1 = z$; abort. Otherwise add $(1^n, y_1)$ to L. If $x \neq 1^n$, query $y = \pi'(x)$ and check whether $y = y_1$. If no add (x, y) to L. Otherwise add $(x, 1^n)$ to L. Clearly the probability that B' aborts at this stage is $1/(2^k - |L|)$. As B_1 makes only n^{c_1} number of queries, the above probability is negligible.

Suppose after all the queries, $B_1(z)$ outputs z'. Select one element x' uniformly at random from $\{0,1\}^n \setminus L_x$, add (x', z') to L and simulate $B_2(z, z', x')$. If B_2 makes an oracle query x ($x \neq 1^n$) s.t $\pi'(x) = z'$ compute $y' = \pi'(x')$. If $x' = 1^n$; $y' = 1^n$. Add (x, y') to L and proceed with y' as the answer. Otherwise if B_2 queries 1^n, select one element y_1 uniformly at random from $\{0,1\}^n \setminus L_y$. If $y_1 = z$; abort. Otherwise add $(1^n, y_1)$ to L. For all other query x compute $y = \pi(x)$. If $y = y_1$ reset $y = 1^n$. Add (x, y) to L.

Suppose B_2 returns x_1. Clearly $x_1 \neq 1^n$ as B' aborted whenever 1^n was queried and the result of the sampling was z. If $\pi'(x_1) = z'$ return x' from the list. else return x_1.

It is easy to check that, while answering the oracle queries B' simulates the random permutation. Moreover, if B_2 returns a correct answer so does B, except the case when B' aborts (when challenge is 1^n or B has queried 1^n and the result of the sampling was z). Probability that randomly chosen challenge is equal to 1^n

is $1/2^n$. On the other hand, Probability that while sampling for the image of 1^n, z was picked is at most $\frac{1}{2^n-|L|} \leq \frac{1}{2^n-n^{c_1}}$ for some fixed constant c. So Probability that B' aborts is negligible. So Probability that B' inverts a randomly chosen z is at least $\frac{1}{n^c-negl(n)}$. This is a contradiction to the fact that a π' is hard to invert. Hence the lemma follows. □

Now, we can claim that TDP^T is one way even relative to G.

Lemma 3.

$$Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z] \leq negl(n),$$

where $x \leftarrow_R \{0,1\}^n$ *and* $(pk, td) \leftarrow Tdg(1^n)$.

Proof. As $F_T(pk, .)$ is a permutation chosen uniformly at random from a set of exponential size, $F_T(pk, .)$ is computationally indistinguishable from a random permutation. So if G was not there, TDP^T was clearly one way. Next we shall prove that TDP^T is one-way even when adversary has access to G. By the property of $PSS_H^{f_{pk}}$, there can be at most one $r \in \{0,1\}^{k_0}$ such that $PSS_H^{f_{pk}}(0||r) = z$. So Probability that G selects that corresponding r is $1/2^{k_0}$ which is negligible for $k_0 = \omega(\log n)$. On the other hand , if G does not select that particular r, by Lemma 2, a random permutation and hence $F_T(pk, .)$ (being computationally indistinguishable from a random permutation) is hard to invert . Hence

$$\begin{aligned}
&Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z] \\
&= Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z | G(\cdot) = F_T(td, z)].Pr[G(\cdot) = F_T(td, z)] \\
&+ Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z | G(\cdot) \neq F_T(td, z)] \cdot Pr[G(\cdot) \neq F_T(td, z)] \\
&\leq 1/2^{k_0} + (2^{k_0} - 1)/2^{k_0}.(negl(n)) \\
&= negl(n).
\end{aligned}$$

□

Using Lemma 1 and Lemma 3, we get the main result of this section as follows

Theorem 1. *There is no blackbox reduction of Security under no message attack of Probabilistic Signature Scheme with superpolynomial randomness space from Oneway Trapdoor Permutations.*

5 No Blackbox Reduction from an Ideal Trapdoor Permutation

The following theorem states that there is no adversary that can break the security of the TDP^T using any adversary (in black-box way) breaking $PSS_H^{TDP^T}$ by chosen message attack when TDP^T is an ideal permutation.

Theorem 2. *There is no black-box reduction from a family of ideal trapdoor permutations to the existential unforgeability against chosen message attack of the PSS signature scheme.*

Like the previous section, we shall construct a oracle G such that there exist a PPTM B such that B^G can forge PSS although TDP^T is secure even relative to G. We define, T and TDP^T as in section 4.

Definition of G The oracle G works as follows. On input the description of the hash function triplet $H = (h, g_1, g_2)$, and the security parameter n, it selects $t = \max(|H|, n)$ messages $m_1, m_2, \cdots m_t$ uniformly at random from $\{0,1\}^* \setminus \{0\}$ and outputs them as a set of challenge messages. G expects valid and *distinct* signatures of all the messages. G also keeps a list (initially empty) of description of input hash functions, the challenge messages and the forgery it returns. If the description of the hash matches then G outputs the same challenge messages. If it gets valid signatures (as described below) then it outputs the previously returned forgery from the list.

Once it receives the messages and the signatures $(m_1, m_2, \cdots, m_t, \sigma_1, \sigma_2, \cdots, \sigma_t)$, G checks for the following conditions.

1. $\sigma_1, \cdots, \sigma_t$ are valid signatures for $m_1, \cdots, m_t$. Recover $r_1, \cdots, r_t$ such that,
$$PSS_H^{f_{pk}}(m_1\|r_1) = f_{pk}(\sigma_1), \cdots, PSS_H^{f_{pk}}(m_t\|r_t) = f_{pk}(\sigma_t).$$
2. $\sigma_i \neq \sigma_j$ (or equivalently $PSS_H^{f_{pk}}(m_i\|r_i) \neq PSS_H^{f_{pk}}(m_j\|r_j)$) for all $1 \leq i < j \leq t$.
3. $\{PSS_H^{f_{pk}}(m_1\|r_1), \cdots, PSS_H^{f_{pk}}(m_t\|r_t)\} \cap Y_{f_{pk}}^{PSS_H}(r_1, \cdots, r_t) = \emptyset$ where
$$Y_{f_{pk}}^{PSS_H}(r_1, \cdots, r_t) = \{f_{pk}(x) | \exists i,\ 1 \leq i \leq t, \\ PSS_H^{f_{pk}}(m_i\|r_i) \textit{ makes the oracle query } x\}.$$

If all the above conditions are satisfied then G chooses one r uniformly at random from $\{0,1\}^{k_0}$ and returns $f_{pk}^{-1}(PSS_H^{f_{pk}}(0\|r))$. Here pk is the public key generated by $Tdg(1^k)$.

G breaks the security of $PSS_H^{TDP^T}$

Lemma 4. *There is a PPTM B^G that can mount existential forgery by chosen message attack on PSS with overwhelming probability.*

Proof. The goal of B^G is to either generate a forgery on it's own or use the `sign` oracle to get signatures of $m_1, \cdots, m_t$ such that Condition 1, Condition 2 and 3 get satisfied. Then B^G can use output of G to produce forgery for the message 0. We describe two constructions of B^G depending on size of the randomness space or k_0.

Algorithm 1 B^G : Phase-I

1: $r_i^k \leftarrow_R \{0,1\}^{k_0} : 1 \leq i \leq t, 1 \leq k \leq t$
2: $V = \{PSS_H^{f_{pk}}(m_i \| r_i^k) : 1 \leq i \leq t, 1 \leq k \leq t\}$
3: $Y = \{f_{pk}(x) | \ \exists i,k,\ 1 \leq i \leq t,\ 1 \leq k \leq t$
$s.\ t.\ PSS_H^{f_{pk}}(m_i, r_i^k)$ *makes oracle query* $f_{pk}(x)\}$
4: **if** $V \cap Y \neq \emptyset$ **then**
5: Output Direct Forgery
6: **end if**

Case I: $k_0 = \mathcal{O}(\log n)$: In this case B^G precomputes $PSS_H^{f_{pk}}(m_i \| r)$ for all $r \in \{0,1\}^{k_0}$ and $i = 1 \cdots t$ and checks whether the Condition 3 from Section 5 would get satisfied or not for any possible choice of r by the `Sign` oracle. If not B can find some m_i, m_j, r_i, r_j, x such that

$$PSS_H^{f_{pk}}(m_i \| r_i) = f_{pk}(x),$$

where $PSS_H^{f_{pk}}(m_j \| r_j)$ makes the oracle query $f_{pk}(x)$. In this case B can easily produce the forged signature x for the message m_i.
Otherwise to take care of Condition 2, B^G calls the `Sign` oracle to get valid signatures for message m_i's one by one for $i = 1$ to t. After receiving the i^{th} signature σ_i it always recovers the randomness r_i and checks whether

$$PSS_H^{f_{pk}}(m_i \| r_i) = PSS_H^{f_{pk}}(m_j \| r_i)$$

for some $i < j \leq t$. Because of Observation 1 it is sufficient to check with the fixed r_i for collision detection purposes. If the above condition gets true again B^G can readily output a forged signature for message m_j as σ_i. Otherwise, B^G ends up with $\sigma_1, \cdots, \sigma_t$ such that all the three conditions in Section 5 are satisfied. So B^G can easily use G to produce a forgery for the message 0. Hence B^G succeeds to forge PSS with probability 1.

Case II: $k_0 = \omega(\log n)$: In this case the randomness space is of superpolynomial size, hence B^G cannot precompute all the possible outputs of $PSS_H^{f_{pk}}(m \| \cdot)$ even for a single message m. However, we observe that the "no collision" requirement or Condition 2 can easily be taken care of by a technique similar to the previous one. To take care of Condition 3, we adopt a sampling procedure. B^G works in two phases. In *Phase-I*, B samples some random r's from $\{0,1\}^{k_0}$ uniformly and simulate the signing procedure by the real `Sign` oracle that would be queried in *Phase-II*. Then the probabilities that Condition 3 gets satisfied in Phase-I or in Phase-II are essentially the same. We set our parameters such a way, with high probability either Condition 3 does not hold in Phase I (hence direct forgery) or it holds in Phase-II (forgery via oracle G, provided Condition 2 holds).

Algorithm 2 B^G : Phase-II

1: **for** $i = 1$ to t **do**
2: $\quad \sigma_i^1 \leftarrow \texttt{Sign}(m_i), \cdots, \sigma_i^t \leftarrow \texttt{Sign}(m_i)$
3: $\quad \Sigma_i = \{\sigma_i^1, \cdots, \sigma_i^t\}$
4: $\quad$ Recover $r_i^1, \cdots, r_i^t$ from $\sigma_i^1, \cdots \sigma_i^t$ using `Verify`.
5: $\quad$ **for** $j = i + 1$ to t **do**
6: $\quad\quad$ **if** $PSS_H^{f_{pk}}(m_i \| r_i^k) == PSS_H^{f_{pk}}(m_j \| r_i^k)$ *for some* $1 \leq k \leq t$ **then**
7: $\quad\quad\quad$ Output Direct Forgery (m_j, σ_i^k)
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **for** $k = 1$ to t **do**
11: $\quad\quad X_{i,k} \leftarrow \{x | PSS_H^{f_{pk}}(m_i \| r_i^k)$ *makes oracle query* $f_{pk}(x)\}$
12: $\quad\quad$ **if** $\sigma_i^k \in X_{i,k}$ **then**
13: $\quad\quad\quad \Sigma_i \leftarrow \Sigma_i \setminus \{\sigma_i^k\}$
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: $\quad$ **if** $\Sigma_i = \emptyset$ **then**
17: $\quad\quad$ Output $\perp$
18: $\quad$ **end if**
19: $\quad$ Pick any $\sigma_i \in \Sigma_i$
20: **end for**
21: **if** $\sigma_1, \cdots, \sigma_t$ satisfy Condition 1, Condition 2 and Condition 3 from Section 5 **then**
22: $\quad$ Output forgery via G
23: **else**
24: $\quad$ Output $\perp$
25: **end if**

Success Probability of B^G in Case II : In Line 21 of Algorithm 2, Condition 1 and Condition 2 are always satisfied. So B^G can abort only in two ways.

1. In Line 16 of Algorithm 2, Σ_i becomes empty for some i, $1 \leq i \leq t$.
2. In Line 21 of Algorithm 2, Condition 3 gets violated. $r_1, \cdots, r_t$ be the random strings recovered from $\sigma_1, \cdots, \sigma_t$. Violation of Condition 3 over here implies there exists some i, j, $1 \leq i, j \leq t$, $i \neq j$ such that

$$PSS_H^{f_{pk}}(m_i \| r_i) = f_{pk}(x),$$

where $PSS_H^{f_{pk}}(m_j \| r_j)$ makes the oracle query x.

Moreover, in both the cases no forgery was found in Algorithm 1.

Let us consider the case where for some i, Σ_i is empty. It implies for some i, for all $k = 1, \cdots, t$, $\sigma_i^k \in X_{i,k}$ and hence was removed from Σ_i . Fix some i . Let us call the set of r for which $PSS_H^{f_{pk}}(m_i, r) = f_{pk}(x)$ and x was queried while computing $PSS_H^{f_{pk}}(m_i, r)$ as BAD. Suppose

$$Pr_r[r \in \text{BAD}] = \theta.$$

Now the event $\Sigma_i = \emptyset$ and no forgery was found in Phase-I implies that the random strings $r^{(i)}$, sampled in Phase 1 were not from the BAD set and all of $r_i^1, r_i^2, \cdots, r_i^t$ was from BAD. As `Sign` and B samples independently, probability of $\Sigma_i = \emptyset$ is $\theta^t(1-\theta)^t \leq 2^{-t}$. Taking union bound over all i, the probability that for some i, Σ_i is empty is at most $t/2^t$.

For the second case, the chosen σ_is were not queried while computing them; rather one σ_i was queried while computing some other σ_j. Recall that maximum number of f_{pk} queries (made by $PSS_H^{f_{pk}}$) while computing one signature is $|H|$. As, for any j $\Sigma_j \leq t$, for each $j = 1, 2, \cdots, t;\ j \neq i$, maximum number of f_{pk} queries made while computing Σ_j is at most $t|H|$. So overall, for all $j \neq i$, total number of f_{pk} queries made by the $PSS_H^{f_{pk}}$ was $t^2|H|$. As, there are $2^{|r|}$ choices of random string, implying $2^{|r|}$ choices for each σ_i^k, and `Sign` runs each time with independent random coins, probability that at least one σ_i^k was from those $t^2|H|$ many f_{pk} queries is at most $\frac{t^4}{2^{k_0}}$.

Hence we get that

$$\begin{aligned}
&\Pr[B^G \rightarrow \perp] \\
&\leq \Pr[\exists i; \Sigma_i = \emptyset] + \Pr[\exists i, j; \sigma_i \in \{\ f_{pk} \text{ queries made while computing } \sigma_j\ \}] \\
&\leq \frac{t}{2^t} + \frac{t^4|H|}{2^{|r|}}
\end{aligned}$$

Putting $t = max(|H|, n), |r| = \omega(\log n)$ and $|H| \leq n^c$ for some constant c, $\Pr[B^G \rightarrow \perp]$ is $negl(n)$. □

G does not break the security of TDP^T

Lemma 5. *For any oracle `PPTM` B and any δ-hard game C (with $t = t(n)$ implicitly defined by C),*

$$Pr[\langle C^{F(pk_1),F(pk_2),\cdots,F(pk_t)}, A^{F(pk_1),F(pk_2),\cdots,F(pk_t),G}\rangle = 1] \leq \delta + negl(n),$$

where $(pk_i, td_i) \leftarrow_R Tdg(1^n)$ for $i = 1, \cdots t$.

Proof. The proof of the above lemma is essentially same as in proof of Lemma 2 in [8], where one argues in the absence of oracle G the claim holds because of computational indistinguishability of f_{pk} from a random permutation. Moreover, Lemma 6 below states the accepting condition of oracle G can only be satisfied with a negligible probability. □

Lemma 6. *Let f be a random permutation on $\{0,1\}^n$ and $c \geq 1$ be a constant, $m_1, \cdots, m_t$ be n-bit values with $t = \max(|H|, n)$. For any oracle `TM` A which makes at most n^c oracle queries, we have (the probability is over randomness of f)*

$$\Pr[A^f \rightarrow (H, x_1, \cdots, x_t)] = \mathtt{negl}(n)$$

where, $|H| \leq n^c$ and the output satisfies the following conditions for some k_0-bit $r_1, \cdots, r_t$

1. $f^{-1}(PSS_H^f(m_1\|r_1)) = x_1, \cdots, f^{-1}(PSS_H^f(m_t\|r_t)) = x_t$.
2. $f^{-1}(PSS_H^f(m_i\|r_i)) \neq f^{-1}(PSS_H^f(m_j\|r_j))$ *for all* $1 \leq i < j \leq t$.
3. $\{PSS_H^f(m_1\|r_1), \cdots, PSS_H^f(m_t\|r_t)\} \cap Y_f^{PSS_H}(r_1, \cdots, r_t) = \emptyset$, *where*

$$Y_f^{PSS_H}(r_1, \cdots, r_t) = \{f(x) | \exists i, 1 \leq i \leq t, \\ PSS_H^f(m_i\|r_i) \text{ makes the oracle query } x\}.$$

Lemma 6 can proved following the same technique of Lemma 3 of [8]. For completeness we give the following proof.

Proof. Consider any oracle `TM` A, where A^f comes up with an output $(PSS_H, x_1, \cdots, x_t)$ after making n^c oracle queries. Even if A outputs only one x_i, which it did not query to the oracle f, then the probability that the relation $f^{-1}(PSS_H^f(m_i\|r_i)) = x_i$ holds for some r_i is negligible. Let X_f^A, $|X_f^A| = n^c$ denote all the oracle queries made by A^f, i.e.

$$X_f^A = \{x | A^f \text{ makes the oracle query } x\}.$$

Consider any fixed oracle circuit h, $|h| \leq n^c$ and k_0 bit values $r_1, \cdots, r_t$ satisfying condition 2 and 3. Let $X_f^h(r_1, \cdots, r_t) = \{f^{-1}(y) | y \in Y_f^h(r_1, \cdots, r_t)\}$, i.e.

$$X_f^h(r_1, \cdots, r_t) = \{x | \exists i, 1 \leq i \leq t, h^f(m_i\|r_i) \text{ makes the oracle query } x\}$$

and let

$$H(r_1, \cdots, r_t) = \{f^{-1}(h^f(m_1\|r_1)), \cdots, f^{-1}(h^f(m_t\|r_t))\}.$$

Condition 3 states that $f(H(r_1, \cdots, r_t)) \cap f(X_f^h(r_1, \cdots, r_t)) = \phi$, and as f is permutation this is equivalent to

$$H(r_1, \cdots, r_t) \cap X_f^h(r_1, \cdots, r_t) = \phi.$$

Given $X_f^h(r_1, \cdots, r_t)$ and conditioned on h^f satisfies condition 3 for the fixed $r_1, \cdots, r_t$, the set $H(r_1, \cdots, r_t)$ is a random subset of $\{0,1\}^n \setminus X_f^h(r_1, \cdots, r_t)$. If condition 2 is satisfied then $H(r_1, \cdots, r_t) = t$, moreover $|X_f^h(r_1, \cdots, r_t)| \leq t|h| \leq tn^c$. Now the probability that $H(r_1, \cdots, r_t) \subseteq X_f^A$ can be upper bounded as

$$\begin{aligned} \Pr[H(r_1, \cdots, r_t) \subseteq X_f^A] &= \prod_{i=0}^{t-1} \frac{|X_f^A| - |X_f^A \cap X_f^h(r_1, \cdots, r_t)| - i}{2^n - i - |X_f^h(r_1, \cdots, r_t)|} \\ &\leq \Big(\frac{|X_f^A|}{2^n - t - |X_f^h(r_1, \cdots, r_t)|}\Big)^t \\ &\leq \Big(\frac{n^c}{2^n - 2tn^c}\Big)^t. \end{aligned}$$

By taking the union bound over all oracle circuits h, $|h| \leq n^c$ and all possible $r_1, \cdots, r_t$ we can now upper bound the probability that there exists some oracle

circuit PSS_H, and k_0-bit values $r_1, \cdots, r_t$ satisfying condition 2 and 3 such that A^f have queried the x_i values satisfying condition 1 as

$$\sum_{|H|=1}^{n^c} 2^{|H|} \Big(\frac{n^c 2^{k_0}}{2^n - 2tn^c} \Big)^t.$$

As $t = \max(|H|, n)$ and assuming $k_0 < n - c \log n$ for all c and sufficiently large n,[3] we can easily show the above term is $\mathtt{negl}(n)$.

6 No Reduction from Lossy Trapdoor Permutations

Lossy Trapdoor Functions, introduced by Peikert et. al. has gained considerable attention in recent years. In a recent work [16], has proven IND-CPA security of OAEP under Lossy Trapdoor Permutation. Moreover different constructions like IND-CCA secure encryption, which cannot be reduced to standard trapdoor permutation using blackbox techniques, were proven reducible to Lossy Trapdoor Permutations. In this section we show that there is no blackbox reduction of existential unforgeability of PSS against chosen message attack from Lossy Trapdoor Permutations as well. Specifically, Let $LTDP = (S, F, F')$ be a family of Lossy Trapdoor Permutation. We define the output of PSS based on LTDP as $\sigma = f^{-1}(PSS_H(m||r))$ where $(f, f^{-1}) \in F$. Note that, while instantiating PSS by a lossy TDP, we consider the trapdoor permutation to be the injective mode of the TDP.

Theorem 3. *There is no blackbox reduction of existential unforgeability against chosen message attack of Probabilistic Signature Scheme from Lossy Trapdoor Permutations.*

Proof To prove Theorem 3, we need new definitions of the oracles.

Definition of $\mathcal{T}$ $\mathcal{T}$ is defined as a pair (T, T'). Choose $2^n + 1$ permutations $f_0, \cdots, f_{2^n-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^n$. Moreover choose 2^n functions $e_0, \cdots, e_{2^n-1}$ uniformly at random from the set of all functions from $\{0,1\}^n$ to $\{0,1\}^l$.

Oracle T works as follows:

- $T_1(td) \rightarrow g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \rightarrow f_{pk}(y)$ (evaluate)
- $T_3(td, z) \rightarrow f^{-1}_{g(td)}(z)$ (inversion)

On the other hand T' is defined as follows

- $T'(pk, x) = f_{pk}(1^{n-l}||e_{pk}(x))$

[3] If $k_0 = n - c \log n$ for some constant c PSS is trivially insecure as a random n bit string will be a valid signature of message 0 with probability $2^{n - c \log n - n} = \frac{1}{n^c}$

Now we define the $LTDP^{T,T'} = (S, (F, F^{-1}), F')$ as follows

- $S(b)$ If $b = 1$, choose a uniform random $td \leftarrow \{0,1\}^n$ computed $pk = T_1(td)$ and return (pk,td), otherwise choose a uniform random $pk \leftarrow \{0,1\}^n$ and return $(pk, \perp)$.
- $F(pk, y)$ returns $T_2(pk, y)$.
- $F^{-1}(td, z)$ returns $T_3(td, z)$.
- $F'(pk, y)$ returns $T'(pk, x)$.

Lemma 7. *$LTDP^{T,T'}$ implements a secure (n, l) Lossy Trapdoor Permutation when $l = \mathcal{O}(n^{\frac{1}{c}})$ for a positive constant c.*

Proof. Recall that, to show the security of $LTDP^{T,T'}$, we need to argue that for any efficient distinguisher D, $|Pr[D^F = 1] - Pr[D^{F'} = 1]|$ is negligible. Consider a random function $e' : \{0,1\}^n \rightarrow \{0,1\}^l$ and a random permutation $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$. It is easy to check that $\pi(1^{n-l}||e'())$ has the same distribution of a random permutation until a collision in e'. e' being a random function, the collision probability is $q^2/2^l$, which is negligible for $q = \mathcal{O}(n^{c_1}))$ for some constant $c_1 > 0$.

Now using the fact that a function (permutation) chosen uniformly at random from the set of exponentially many functions (permutations) is indistinguishable form a random function (permutation), the lemma follows. □

Definition of G : Intuitively, G will work exactly the same way as in the previous case when the underlying permutation is in injective mode. When the permutation is lossy G can abort instead of returning a forgery. So effectively, when instantiated by the lossy mode G always aborts and in injective mode G aborts if the conditions are not satisfied.

In more detail, G works in the following way. On input the description of the hash functions h, g_1 and g_2, it selects t (to be fixed later) messages $m_1, m_2, \cdots m_t$ uniformly at random from $\{0,1\}^* \setminus 0$ and outputs them as a set of challenge messages. G expects valid and *distinct* signatures of all the messages. G also keeps a list (initially empty) of description of input hash functions, the challenge messages and the forgery it returns. If the description of the hash matches then G outputs the same challenge messages. If it gets valid signatures (as described below) then it outputs the same forgery from the list.

Once it receives the messages and the signatures $(m_1, m_2, \cdots, m_t, \sigma_1, \sigma_2, \cdots, \sigma_t)$, G first checks whether the signatures are valid and distinct.

- $F(pk, \sigma_i) = PSS_H^{f_{pk}}(m_i \| r)$ for some r. This signature verification is to make sure that that calling algorithm has access to signing oracle.
- $\sigma_i \neq \sigma_j$ for all $i \neq j$

If the above two conditions are satisfied then G finds the random strings used in the signatures. Let $r_1, r_2, \cdots, r_t$ be the random strings

– $\{F(pk,\sigma_1), F(pk,\sigma_2), \cdots, F(pk,\sigma_t)\} \cap Y_T = \emptyset$ where

$$Y_T = \{F(pk,x) | \exists i, 1 \leq i \leq t,\ PSS_H^{f_{pk}}(m_i \| r_i) \text{ queries } F(pk,x)\}.$$

Finally G checks whether F is the lossy mode[4], if yes it aborts; otherwise G chooses one r uniformly at random from $\{0,1\}^{k_0}$ and computes the PSS hash of $0\|r$ as $y = 0\|h(0\|r)\|g_1(h(0\|r)) \oplus r\|g_2(h(0\|r))$. Finally it returns the forgery as $(0, F^{-1}(td, y))$.

In order to use G to distinguish the lossy and the injective mode, any distinguisher has to construct a satisfying assignment of G in injective mode. By Lemma 6, it happens with negligible probability and we get the following result.

Lemma 8. *Suppose $k = \mathcal{O}(n^{\frac{1}{c}})$ for a positive constant c. $LTDP^{T,T'}$ implements a secure (n,k) Lossy Trapdoor Permutation even relative to G.*

Existence of a forger B^G for PSS using the injective mode of the LTDP is satisfied by Lemma 4. This completes the proof of Theorem 3. □

7 No Reduction from Hard Games with Inversion

Like [8], our result can also be extended to the hard games with inversions. Informally, in a hard game with bounded inversion $\mathcal{C}$, the adversary is allowed to make polynomial $q(n)$ many inversion queries except on some points defined in the game (for one way game adversary is not allowed to make inversion queries on the challenge she received). Following [8], if we modify G to ask for signatures of $|H| + q(n)$ messages and modify Lemma 6 accordingly, we get the following two theorems.

Theorem 4. *There is no blackbox reduction of security against existential forgery under chosen message attack for PSS from any hard game with polynomial number of inversion queries.*

Theorem 5. *There is no blackbox reduction of security against existential forgery against zero message attack for PSS from an oneway trapdoor permutation, even with polynomial number of inversion queries.*

8 Conclusion

Following the negative results, on generic insecurity of FDH by Dodis et. al [8] and of OAEP by Kiltz and Pietrzak [17] in standard model, we show security of PSS also can not be black box reduced to any property of an ideal trapdoor permutation. Moreover, we also show one can not even hope to achieve security of PSS based on Lossy Trapdoor Permutations. On the contrary recently a secure instantiation of OAEP has been realized based on Lossy Trapdoor Permutations [16].

[4] As description of F can be hardwired in G, G can easily check the mode of F by finding the possible inverses

9 Acknowledgments

Rishi thanks Palash Sarkar for helpful discussions.

References

1. Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, pages 83–107, 2002.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.
4. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *STOC*, pages 209–218, 1998.
5. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
6. Jean-Sébastien Coron. Optimal security proofs for pss and other signature schemes. In *EUROCRYPT*, pages 272–287, 2002.
7. Jean-Sébastien Coron and Avradip Mandal. Pss is secure against random fault attacks. In *ASIACRYPT*, pages 653–666, 2009.
8. Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In *CRYPTO*, pages 449–466, 2005.
9. Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT*, pages 197–215, 2010.
10. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS*, pages 305–313, 2000.
11. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335, 2000.
12. Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pages 126–135, 2001.
13. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.
14. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO*, pages 92–105, 2004.
15. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61, 1989.
16. Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of rsa-oaep under chosen-plaintext attack. In *CRYPTO*, pages 295–313, 2010.
17. Eike Kiltz and Krzysztof Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove oaep secure in the standard model. In *EUROCRYPT*, pages 389–406, 2009.
18. Pascal Paillier. Impossibility proofs for rsa signatures in the standard model. In *CT-RSA*, pages 31–48, 2007.
19. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.
20. Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, pages 334–345, 1998.