A preliminary version of this paper appears in EUROCRYPT 2016. This is the full version.

New Negative Results on Differing-Inputs Obfuscation

Mihir Bellare[1] Igors Stepanovs[2] Brent Waters[3]

June 2015

Abstract

We show that differing-inputs obfuscation (diO) for Turing Machines is impossible to achieve. Our results are: (1) If sub-exponentially secure one-way functions exist then sub-exponentially secure diO for TMs does not exist (2) If in addition sub-exponentially secure iO exists then polynomially secure diO for TMs does not exist.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@eng.ucsd.edu`. URL: `https://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grants CNS-1526801 and CNS-1228890, ERC Project ERCC FP7/615074 and a gift from Microsoft.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `istepano@eng.ucsd.edu`. URL: `https://cseweb.ucsd.edu/~istepano/`. Supported in part by NSF grants CNS-1526801 and CNS-1228890, ERC Project ERCC FP7/615074 and a gift from Microsoft.

[3] Department of Computer Science, University of Texas at Austin, 2317 Speedway, Austin, Texas 78712, USA. Email: `bwaters@cs.utexas.edu`. URL: `https://www.cs.utexas.edu/~bwaters/`. Supported in part by NSF grants CNS-1228599 and CNS-1414082, DARPA SafeWare, a Google Faculty Research award, the Alfred P. Sloan Fellowship, a Microsoft Faculty Fellowship and a Packard Foundation Fellowship.

Contents

1 Introduction

Differing-inputs obfuscation (diO) is a natural extension of indistinguishability obfuscation (iO). It has been conjectured that candidate constructions of iO also met diO. Based on this, diO has been exploited in applications. Garg, Gentry, Halevi and Wichs (GGHW) [27] showed that if something they called "special purpose" obfuscation exists, then diO does not. This has put diO in an ambiguous and contentious position, some people arguing that GGHW is evidence diO does not exist, others saying that perhaps it does and it is special-purpose obfuscation that does not exist. This paper uses a new approach to give powerful evidence that the first camp is right, meaning it is indeed diO that does not exist, by showing this to be true under weaker and more standard assumptions than special-purpose obfuscation. We show (1) If sub-exponentially secure one-way functions exist then sub-exponentially secure diO for TMs does not exist (2) If in addition sub-exponentially secure iO exists then polynomially secure diO for TMs does not exist.

Background. The notion of program obfuscation that is most intuitive and appealing is that an obfuscated program should be no more useful than an oracle for the program itself. Formalized as VBB obfuscation (vbbO), it was shown impossible in the sense that there is no obfuscator that will successfully VBB obfuscate all programs [35, 7]. Further negative results about vbbO were given in [32, 15]. In the face of this, Barak et al. [7] suggested other, weaker notions of obfuscation that appeared not to succumb to their counter-examples and might therefore be achievable. The most prominent were indistinguishability obfuscation (iO) and its extension, differing-input obfuscation (diO). The first asks that obfuscations of functionally equivalent programs are indistinguishable. The second is a natural computational relaxation: even if the programs are not functionally equivalent, as long as it is hard, given the programs, to find an input on which they differ, then the obfuscations of the programs are indistinguishable. The underlying intuition is that if one can find a differing input for the programs, one can clearly distinguish their obfuscations. In iO this is excluded information theoretically, by saying there does not exist such an input, while in diO it is excluded computationally, by saying such an input might exist but is hard to find. On the surface both might appear equally reasonable, since the vbbO negative results do not apply to either. But this turns out not to be true.

These intriguing notions lay dormant for many years, for two reasons. First, that one could not prove these notions unachievable did not mean they were achievable. Second, they seemed quite weak; even if they were achievable, what could one do with them? An answer to the first question came with candidate constructions of iO [26, 6, 42, 29]. An answer to the second came when Sahai and Waters showed how to use iO towards many ends [44]. Since then, applications of iO and diO have ballooned.

In these applications, a crucial role is played by *auxiliary information.* The modern definitions of iO and diO used in these applications [26, 44, 1, 18, 12] consider a program sampler S that spits out a pair $\mathrm{P}_0, \mathrm{P}_1$ of programs *together with associated auxiliary information aux.* The sampler is said to produce functionally equivalent programs if P_0 and P_1 agree on all inputs. The sampler is said to be difference-secure if an adversary given $\mathrm{P}_0, \mathrm{P}_1, aux$ cannot find an input x such that $\mathrm{P}_0(x) \neq \mathrm{P}_1(x)$ except with small probability. The obfuscation game picks a challenge bit b and gives you (the adversary) an obfuscation $\overline{\mathrm{P}}$ of P_b under the obfuscator Obf, *together with aux.* Your task (as the adversary) is to guess b. Obf is called iO-secure if you have small advantage for all samplers producing functionally equivalent programs, and diO-secure if you have small advantage for all difference-secure samplers. Adversaries are always polynomial time, but probabilities referred to as "small" may be sub-exponentially so or negligible. Programs may be TMs or circuits. This leads to a collection of variant notions.

The GGHW result. Let Obf be an obfuscator. GGHW [27] provide a program sampler S for which they show, under certain assumptions, that diO-security of Obf fails, which means that (1) the sampler is difference secure under these assumptions, but (2) there is a way to distinguish the obfuscations under Obf of the two programs returned by the sampler given the auxiliary information. Their approach is to have the sampler first generate a signing and verification key pair (sk, vk) for a signature scheme meeting the standard notion of unforgeability [33]. The program P_1 takes a message m and candidate signature σ and accepts iff σ is a valid signature on m under vk. The program P_0 will take in the same inputs, but it will always reject. Clearly the programs P_0 and P_1 differ exactly on the input pairs (m, σ) where σ is a valid signature of m under vk. Next, the sampler creates a third program P_2 that has hardwired the secret signing key sk and takes as input a (smaller) program $\overline{\mathrm{P}}$. It hashes $\overline{\mathrm{P}}$ using a CRHF to get a message m, and uses sk to get a signature σ on m. It then runs the $\overline{\mathrm{P}}$ on (m, σ) and outputs 1 if $\overline{\mathrm{P}}$ accepts on these inputs. Finally, S creates auxiliary information aux consisting of an obfuscation P_2^* of P_2. This obfuscation is not under the given obfuscator Obf, but under some other assumed "special purpose" obfuscator Obf^* whose role and properties will emerge in the following.

To serve as a counterexample it should both (1) be possible, using the auxiliary information P_2^*, to distinguish between obfuscations under Obf of P_0 and P_1, and (2) be difficult, given $\mathrm{P}_0, \mathrm{P}_1, \mathrm{P}_2^*$, to find an input on which P_0 and P_1 differ. The first property follows trivially from the design. An adversary given the auxiliary information P_2^* and a challenge program $\overline{\mathrm{P}}$ that is either an obfuscation of P_0 or P_1 can distinguish these cases by simply feeding the program $\overline{\mathrm{P}}$ as an input to P_2^*. If $\overline{\mathrm{P}}$ is an obfuscation of P_1 then, when P_2^* runs $\overline{\mathrm{P}}$ on the message and valid signature that P_2^* creates, $\overline{\mathrm{P}}$ will accept. But if $\overline{\mathrm{P}}$ is an obfuscation of P_0, then P_2^* will reject.

In contrast it is much more difficult to establish the second property, namely that it is hard to find an input on which $\mathrm{P}_0, \mathrm{P}_1$ differ *even in the presence of the auxiliary information* P_2^*. The difficulty stems from the latter. In the absence of aux the property follows straightforwardly from the security of the signature scheme, as a differing input is exactly a valid message-signature pair, and would amount to a signature forgery. However, since the obfuscated differentiating program P_2^* has embedded in it the secret signing key sk it is not clear how to prove that it is hard to find signatures in the presence of P_2^*.

Recall that P_2^* was an obfuscation, under some un-specified obfuscator Obf^*, of P_2. GGHW [27] simply conjecture that there exists some obfuscator Obf^* that will hide the secret key sk sufficiently well that it is hard to find a differing input for $\mathrm{P}_0, \mathrm{P}_1$, meaning to find a valid message-signature pair, even given P_2^*. While they were unable to prove this conjecture under any standard obfuscation definitions such as iO or even vbbO, they were able to partially justify their conjecture with a heuristic analysis. Their analysis replaces the adversary's access to the obfuscated program P_2^* with an oracle that performs the same functionality. In this world the adversary no longer has direct access to an object containing sk and GGHW are able to demonstrate differing inputs security of S by a fairly straightforward reduction to the underlying security of the signature scheme.

The GGHW result certainly creates significant questions regarding the use of diO. Arguably, the primary reason for using the diO security definition over vbbO is that no impossibility results like those of [35, 7, 32, 15] are known for diO. However, if the GGHW conjecture holds, then this is no longer true and the perceived benefit of diO versus vbbO is significantly reduced. (The benefit is not eliminated, since even if there exist functionalities that cannot be diO obfuscated, it is still possible that there are functionalities that can be diO obfuscated but not VBB obfuscated.) At the same time, the heuristic used to justify the GGHW counterexample is itself much stronger than assuming diO — namely their analysis relies on modeling the differentiating obfuscated program as an oracle.

Our Approach. We introduce a new approach to proving the impossibility of diO. In contrast to the prior work, we analyze our sampler under *concrete assumptions* that replace the GGHW conjecture. We now explain the intuition behind our approach as well as the obstacles we had to overcome.

Let Obf be an obfuscator that we assume, towards a contradiction, is diO-secure. At the highest level our approach is similar to GGHW. We build a program sampler S that produces programs $\mathrm{P}_0, \mathrm{P}_1$ and auxiliary information P_2^* consisting of an obfuscation of a program P_2 under an obfuscator Obf^*. As in GGHW, the sampler generates a signing and verification key pair (sk, vk) for an underlying signature scheme DS, and program P_0 always rejects. Likewise, P_1 takes as input a candidate message-signature pair (m, σ) and checks its validity under the signature verification program $\mathsf{DS.Ver}$ with key vk. The auxiliary information continues to be the obfuscation P_2^*, under an obfuscator Obf^*, of a program P_2, where P_2 hardwires the secret signing key sk. P_2 takes as input a program $\overline{\mathrm{P}}$ of a certain maximum length, and uses $m = \overline{\mathrm{P}}$ as the message it signs, and runs $\overline{\mathrm{P}}$ on m and the signature, accepting if this accepts. The important difference now however is that Obf^* is not some new type of obfuscator as in GGHW. Rather Obf^* is *assumed to be only an iO-secure obfuscator.*

It continues to be easy, using the auxiliary information P_2^*, to distinguish between obfuscations under Obf of P_0 and P_1. The main issue is to prove that it is difficult, given $\mathrm{P}_0, \mathrm{P}_1, \mathrm{P}_2^*$, to find an input on which P_0 and P_1 differ. The hurdle here continues to be the same, namely that the auxiliary information program P_2^* embeds the secret signing key sk. This precludes reducing to the security of the signature scheme in an obvious way. To prove security we will show that it is computationally difficult to generate a signature on any message. We do this via a hybrid argument that steps through every possible message one by one. Since our hybrid steps through the entire message space we base our security on assumptions of sub-exponential hardness.

To execute our strategy we will replace the generic signature scheme of GGHW with a special type of puncturable signature scheme that we call a *consistent puncturable* signature scheme. Given a "master" secret key sk, it should be possible to create a punctured version sk_{m^*} of the key, for a given message m^*, that can be used to sign any message $m \neq m^*$ but even given which it is hard to produce a signature on m^*. So far this is a special type of policy-based [9], functional [19] or delegatable [5] signatures, these themselves analogues of the notions of puncturable, constrained and functional PRFs [17, 39, 19]. The additional consistency requirement is that the signatures of $m \neq m^*$ produced under the master key and the punctured key should be the same. Note that only deterministic puncturable signature schemes can be consistent, but the former is not a sufficient condition. We show in Section 3 that such signature schemes can be built from iO and one-way functions. While making a standard signature scheme deterministic is trivial via the use of PRFs, our challenge is making the punctured and master versions of the key produce consistent signatures.

Our hybrid now proceeds as follows. We step through each program (message) P^* and show that it is computationally difficult to produce a signature on P^*. We do this by first replacing the obfuscation of P_2 with an obfuscation of a program $\mathrm{P}_{2,\mathrm{P}^*}$ that works as follows. On all inputs $\mathrm{P} \neq \mathrm{P}^*$ the program $\mathrm{P}_{2,\mathrm{P}^*}$ behaves as P_2 with the exception that it uses a punctured version of the signing key sk_{P^*}. On input P^* its output is hardwired to be whatever the output of $\mathrm{P}_2(\mathrm{P}^*)$ was. We observe that if indistinguishability obfuscation holds, then no poly-time attacker can distinguish between obfuscations of programs P_2 and $\mathrm{P}_{2,\mathrm{P}^*}$. This follows since the two programs share the same output on every input. On every $\mathrm{P} \neq \mathrm{P}^*$ the master and punctured keys will produce the same signature that they feed into P, and on input P^* program $\mathrm{P}_{2,\mathrm{P}^*}$ is hardwired to behave the same as P_2. Since it is hard to distinguish between obfuscations of these two programs, it should be no easier to output a signature on message P^* when P_2 is obfuscated to get the auxiliary information *aux* than it is when $\mathrm{P}_{2,\mathrm{P}^*}$ is obfuscated. However, in the latter case the security of the puncturable signature scheme guarantees this is hard.

Note that since we assumed a diO-secure obfuscator Obf to start our proof by contradiction, an iO-secure obfuscator, which we use both directly and to build consistent punctured signatures, is provided for free and is not an extra assumption. This means the only assumption we need is a sub-exponentially hard one-way function. More precisely, this is the case for sub-exponential diO, while for polynomial diO the iO assumption will be an extra one.

While the text above outlines our main approach, there are several important factors that still must be taken into account. First, we notice that P_1 should be capable of verifying a signature on a message that is an obfuscation of P_1 and thus longer than P_1 itself. For this reason we need to view P_0 and P_1 as Turning Machines (TMs) that can process inputs longer than their own descriptions.

Next, our complexity leveraging argument requires that the advantage ϵ of any PT attacker on the signature scheme multiplied by the message space be negligible. To satisfy this using sub-exponential hardness assumptions we must use a verification key vk that is larger than the programs $\mathrm{P}_0, \mathrm{P}_1$. However, this creates a circularity problem under the obvious strategy of having P_1 actually contain vk to verify the messages! We circumvent this issue by the use of a target collision-resistant (TCR) hash function (also called a UOWHF) that hashes a separate verification program as follows. We construct a program $\mathrm{P}_{\mathsf{ver}}$ that takes as input a candidate message-signature pair (m, σ) and uses an embedded verification key vk to either accept or reject it. Now P_1 takes $\mathrm{P}'_{\mathsf{ver}}$ as an additional input and uses it to check the candidate message-signature pairs, rather than storing vk and performing the verification itself. P_1 hardwires the hash h of $\mathrm{P}_{\mathsf{ver}}$ under a TCR hash function, and rejects unless the hash h' of $\mathrm{P}'_{\mathsf{ver}}$ matches its hardwired hash h. This ensures that only $\mathrm{P}_{\mathsf{ver}}$ can be used to verify the signatures. We analyze security by adding a hybrid step at the beginning using the UOWHF security. We emphasize that the argument using our UOWHF is outside of the complexity leveraging part of our hybrid.

The above is a very high-level description, and the devil is in the details that the body of the paper sorts out. The circularity issues, summarized via Fig. 5, have to be dealt with very carefully. A critical element of dealing with them is that *different primitives are run with different values of the security parameter.* Thus, while the convention is that the security parameter in a proof remains λ throughout, our constructions will feature $n(\lambda)$ as the security parameter in certain places, with n a polynomial that is carefully defined based on other parameters. Another subtlety is that the success of this program depends on the details of how sub-exponential security is defined. Specifically (cf. Section 2) we use "uniform" rather than "pointwise" definitions in the language of [8]. The latter showed them equivalent in the usual setting of negligible functions but they are not known to be equivalent in the sub-exponential setting.

Discussion. Our assumptions and conclusions both involve sub-exponential hardness and one might ask about the validity of such assumptions and the value of such conclusions. Empirical evidence, at least, says that when problems are hard, they are sub-exponentially hard. Natural problems do not appear to be polynomially but not sub-exponentially hard except in rare cases [3]. Indeed sub-exponential hardness is frequently assumed in cryptography, especially recently [34, 30, 23]. In particular it is unlikely that polynomially-secure diO exists but sub-exponentially secure diO does not, so ruling out the latter is significant in terms of evidence against diO. Similarly it is unlikely that polynomially secure OWFs exist but sub-exponentially secure ones do not, so assuming the latter is reasonable.

Differing-inputs obfuscation has proven to be a powerful tool using which we have built new primitives. In some cases it has later been possible to reduce the assumption to iO or other diO variants, but sometimes at the cost of weakening the conclusion and usually at the cost of increased complexity and difficulty. Thus diO for circuits is used in [1, 18] to elegantly achieve iO for TMs with unbounded input, adaptively-secure FE (Functional Encryption) and extractable witness encryption. The assumption for TM iO was reduced to circuit iO in [22, 14, 40] but the conclusion was weaker. The original result is shown in [37] under public-coin diO. Adaptively-secure FE from iO did emerge but the solutions were more complex than the ones from diO [46, 2]. Differing input obfuscation is used as a tool in [12], via the result of Boyle, Chung and Pass [18], to give hardcore functions with polynomially-many output bits from any injective one-way function and iO, and is used as an assumption to extend this result to arbitrary one-way functions. It is used similarly as a tool in [21]. It is used as an assumption in a result in [28]. All this motivates understanding whether or not diO is achievable.

Related work. Bolyle, Chung and Pass [18] show that iO implies diO for samplers outputting circuits that differ on only polynomially-many inputs. Our counter-examples and results do not apply to this type of diO. They also do not apply to public-coin diO [37].

We get consistent puncturable signatures from OWFs and iO, which in our context effectively means from OWFs since our proof assumes diO towards a contradiction and thus gets iO for free. Our definition of consistent puncturable signatures is novel, but our construction follows Sahai-Waters signatures [44]. Splittable signatures [40] also imply consistent puncturable signatures; they are built based on an injective PRG and iO. Injective PRGs are not known to be implied by OWFs so the assumption is stronger than ours. However, [16] build injective OWFs from OWFs and iO, and also say that, due to an observation of Boyle at al. [18], the injective PRG of [40] can be replaced with an injective OWF. By this route one can get consistent puncturable signatures from OWFs and iO. However our construction is direct, substantially simpler and self contained. Consistent puncturable signatures can also be constructed from constrained verifiable PRFs [25, 24]. The latter are achievable from κ-Multilinear DDH assumption. In our context, this would be an additional assumption since it is not known to be implied by diO.

Some of the prior work focuses on constructing digital signature schemes with properties that are similar to the ones we require above. The proposed primitives include: functional signatures [19], policy-based signatures [9] and operational signatures [4], the latter subsuming the preliminary work on delegatable signatures [5]. However, none of the proposed constructions of these primitives satisfy the consistency requirement which requires that the master and punctured signing keys produce the same signatures for all messages except for the punctured message, and which is crucial for our impossibility result.

Bitansky et al. [13] show that if iO exists then auxiliary-input extractable OWFs do not. Under the stronger assumption of public-coin diO for TMs, [20] obtain a stronger conclusion, in which auxiliary-input extractable OWFs are ruled out with auxiliary input that does not depend on the OWF.

2 Preliminaries

Notation. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of non-negative integers. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation. If $x \in \{0,1\}^*$ is a string then $|x|$ denotes its length. If $x \in \{0,1\}^*$ is a string and $\ell \in \mathbb{N}$ such that $|x| \leq \ell$ then $\langle x \rangle_\ell$ denotes the string of length ℓ that is built by padding x with leading zeros. If X is a finite set, we let $x \leftarrow\$ X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. "PT" stands for "polynomial-time," whether for randomized algorithms or deterministic ones. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \leftarrow\$ A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$. We say that $f\colon \mathbb{N} \to \mathbb{R}$ is negligible if for

```
Game $\mathrm{OW}_{\mathsf{F}}^{\mathcal{F}}(\lambda)$
$fk \leftarrow_{\$} \mathsf{F.Kg}(1^\lambda)$
$x \leftarrow_{\$} \mathsf{F.In}(\lambda)$
$y \leftarrow \mathsf{F.Ev}(1^\lambda, fk, x)$
$x' \leftarrow_{\$} \mathcal{F}(1^\lambda, fk, y)$
$y' \leftarrow \mathsf{F.Ev}(1^\lambda, fk, x')$
Return $(y = y')$
```

```
Game $\mathrm{TCR}_{\mathsf{H}}^{\mathcal{H}}(\lambda)$
$(x_0, st) \leftarrow_{\$} \mathcal{H}_1(1^\lambda)$
$hk \leftarrow_{\$} \mathsf{H.Kg}(1^\lambda)$
$x_1 \leftarrow_{\$} \mathcal{H}_2(1^\lambda, st, hk)$
$h_0 \leftarrow \mathsf{H.Ev}(1^\lambda, hk, x_0)$
$h_1 \leftarrow \mathsf{H.Ev}(1^\lambda, hk, x_1)$
$\mathsf{win}_0 \leftarrow (x_0 \neq x_1)$
$\mathsf{win}_1 \leftarrow (h_0 = h_1)$
Return $(\mathsf{win}_0 \wedge \mathsf{win}_1)$
```

```
Game $\mathrm{PPRF}_{\mathsf{G}}^{\mathcal{G}}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$ ; $gk \leftarrow_{\$} \mathsf{G.Kg}(1^\lambda)$
$b' \leftarrow_{\$} \mathcal{G}^{\mathrm{CH}}(1^\lambda)$ ; Return $(b = b')$

$\mathrm{CH}(x^*)$
$gk^* \leftarrow_{\$} \mathsf{G.PKg}(1^\lambda, gk, x^*)$
If $b = 1$ then $r^* \leftarrow \mathsf{G.Ev}(1^\lambda, gk, x^*)$
else $r^* \leftarrow_{\$} \mathsf{G.Out}(\lambda)$
Return $(gk^*, r^*)$
```

Figure 1: Games defining one-wayness of function family F, target collision-resistance of function family H and puncturable-PRF security of function family G.

every positive polynomial p, there exists $\lambda_p \in \mathbb{N}$ such that $f(\lambda) < 1/p(\lambda)$ for all $\lambda \geq \lambda_p$. We use the code based game playing framework of [11]. (See Fig. 1 for an example.) By $\mathrm{G}^{\mathcal{A}}(\lambda)$ we denote the event that the execution of game G with adversary $\mathcal{A}$ and security parameter λ results in the game returning true.

Uniform and pointwise security definitions. There are two common ways to formalize security definitions – by using different order of quantification. Let GAME be a security game, and let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{game}}(\lambda)$ be the advantage of a PT adversary $\mathcal{A}$ winning in this game with security parameter λ. Consider the following two alternative definitions of sub-exponential security. A *uniform* definition requires that there is a constant $0 < \epsilon < 1$ such that for every PT adversary $\mathcal{A}$ there exists $\lambda_{\mathcal{A}} \in \mathbb{N}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{game}}(\lambda) \leq 2^{-\lambda^\epsilon}$ for all $\lambda \geq \lambda_{\mathcal{A}}$. A *pointwise* definition requires that for every PT adversary $\mathcal{A}$ there exist $0 < \epsilon < 1$ and $\lambda_{\mathcal{A}} \in \mathbb{N}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{game}}(\lambda) \leq 2^{-\lambda^\epsilon}$ for all $\lambda \geq \lambda_{\mathcal{A}}$. These definitions differ in the order of quantification between ϵ and $\mathcal{A}$. In this work, we use uniform security definitions. For the case of polynomial security, Bellare [8] proved that uniform and pointwise definitions are equivalent. It is not known whether the equivalence also holds for the above definitions of sub-exponential security.

Circuits and Turing Machines. We say that P is a program if it is either a circuit or a Turing Machine (TM), and we denote the size of its binary representation by $|\mathrm{P}|$. We assume that any program P takes a single input string x; if P is defined to take multiple inputs $x_1, \ldots$ then running P on an input x is implicitly assumed to parse $(x_1, \ldots) \leftarrow x$ and run $\mathrm{P}(x_1, \ldots)$.

We say that circuits $\mathrm{C}_0, \mathrm{C}_1$ are functionally equivalent, written $\mathrm{C}_0 \equiv \mathrm{C}_1$, if they have the same number of inputs $\ell \in \mathbb{N}$ and if $\mathrm{C}_0(x) = \mathrm{C}_1(x)$ holds for all $x \in \{0,1\}^\ell$. We say that TMs $\mathrm{M}_0, \mathrm{M}_1$ are functionally equivalent, and denote it by $\mathrm{M}_0 \equiv \mathrm{M}_1$, if both $\mathrm{M}_0(x)$ and $\mathrm{M}_1(x)$ halt on all $x \in \{0,1\}^*$ and if $\mathrm{M}_0(x) = \mathrm{M}_1(x)$ for all $x \in \{0,1\}^*$.

If M is a TM and $t \in \mathbb{N}$ then $y \leftarrow \mathsf{UTM}_{\mathrm{M}}^{t}(x_1, \ldots)$ denotes running M on inputs $x_1, \ldots$ and assigning the output to y; if $\mathrm{M}(x_1, \ldots)$ does not halt within t steps, then $\mathsf{UTM}_{\mathrm{M}}^{t}(x_1, \ldots)$ returns 0. If M is a TM and $x \in \{0,1\}^*$ is a string such that M halts on input x, we use $\mathsf{time}(\mathrm{M}, x)$ to denote the number of steps that are required for it to halt.

Let P be any circuit or any TM that halts on all inputs. For any $s \in \mathbb{N}$ such that $|\mathrm{P}| \leq s$ let $\mathsf{Pad}_s(\mathrm{P})$ denote P padded to have size s, meaning that $\mathsf{Pad}_s(\mathrm{P})$ and P are of the same type (i.e. both are circuits or TMs) and $\mathsf{Pad}_s(\mathrm{P}) \equiv \mathrm{P}$. We assume that P can be padded to any size larger or equal to $|\mathrm{P}|$.

Function families. A family of functions F specifies PT algorithms $\mathsf{F.Kg}$ and $\mathsf{F.Ev}$, where $\mathsf{F.Ev}$ is deterministic. Assocated to F is a collection if input sets $\mathsf{F.In}$ and a collection of output sets $\mathsf{F.Out}$, defining all valid inputs and outputs for each of security parameters. Key generation algorithm $\mathsf{F.Kg}$ takes 1^λ to return a key fk. Evaluation algorithm $\mathsf{F.Ev}$ takes $1^\lambda, \mathit{fk}$ and an input $x \in \mathsf{F.In}(\lambda)$ to return $\mathsf{F.Ev}(1^\lambda, \mathit{fk}, x) \in \mathsf{F.Out}(\lambda)$.We say that F is injective if the function $\mathsf{F.Ev}(1^\lambda, \mathit{fk}, \cdot)\colon \mathsf{F.In}(\lambda) \to \mathsf{F.Out}(\lambda)$ is injective for all $\lambda \in \mathbb{N}$ and all $\mathit{fk} \in [\mathsf{F.Kg}(1^\lambda)]$.

Puncturable function families. A *puncturable* function family G specifies (beyond the usual algorithms) additional PT algorithms $\mathsf{G.PKg}$ and $\mathsf{G.PEv}$, where $\mathsf{G.PEv}$ is deterministic. Punctured key generation algorithm $\mathsf{G.PKg}$ takes 1^λ, a key $gk \in [\mathsf{G.Kg}(1^\lambda)]$ and a target input $x^* \in \mathsf{G.In}(\lambda)$ to return a "punctured" key gk^*. Punctured evaluation algorithm $\mathsf{G.PEv}$ takes $1^\lambda, gk^*$ and an input $x \in \mathsf{G.In}(\lambda)$ to return $\mathsf{G.PEv}(1^\lambda, gk^*, x) \in \mathsf{G.Out}(\lambda)$. The correctness condition requires that $\mathsf{G.PEv}(1^\lambda, gk^*, x) = \mathsf{G.Ev}(1^\lambda, gk, x)$ for all $\lambda \in \mathbb{N}$, all $gk \in [\mathsf{G.Kg}(1^\lambda)]$, all $x^* \in \mathsf{G.In}(\lambda)$, all $gk^* \in [\mathsf{G.PKg}(1^\lambda, gk, x^*)]$ and all $x \in \mathsf{G.In}(\lambda) \setminus \{x^*\}$.

One-way functions. Consider game OW of Fig. 1 associated to a function family F and an adversary $\mathcal{F}$, where $\mathsf{F.In}(\lambda)$ is required to be finite for all $\lambda \in \mathbb{N}$. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathcal{F}}(\lambda) = \Pr[\mathrm{OW}^{\mathcal{F}}_{\mathsf{F}}(\lambda)]$. Let $\delta\colon \mathbb{N} \to \mathbb{R}$ be any function. We say that F is δ-OW-secure if for every PT adversary $\mathcal{F}$ there exists $\lambda_{\delta,\mathcal{F}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathcal{F}}(\lambda) \le \delta(\lambda)$ for all $\lambda \ge \lambda_{\delta,\mathcal{F}}$. We say that F is sub-exponentially OW-secure if it is $2^{-(\cdot)^\epsilon}$-OW-secure for some $0 < \epsilon < 1$.

Target collision-resistant functions. Consider game TCR of Fig. 1 associated to a function family H and an adversary $\mathcal{H}$. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda) = \Pr[\mathrm{TCR}^{\mathcal{H}}_{\mathsf{H}}(\lambda)]$. Let $\delta\colon \mathbb{N} \to \mathbb{R}$ be any function. We say that H is δ-TCR-secure if for every PT adversary $\mathcal{H}$ there exists $\lambda_{\delta,\mathcal{H}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda) \le \delta(\lambda)$ for all $\lambda \ge \lambda_{\delta,\mathcal{H}}$. We say that H is sub-exponentially TCR-secure if it is $2^{-(\cdot)^\epsilon}$-TCR-secure for some $0 < \epsilon < 1$. Target collision-resistant hash functions were introduced by Naor and Yung [41] under the name of Universal One-Way Hash Functions (UOWHF). [10] redefined the corresponding security notion under the name of *target collision-resistance*.

TCR-secure function families can be built from one-way functions, by combining the following results. First, [43, 36] (see also [38]) proposed constructions of TCR-secure *compression function* families with fixed input and output lengths. More formally, they show how to build a function family H' such that $\mathsf{H}'.\mathsf{In}(\cdot) = \{0,1\}^{p_{\mathsf{in}}(\cdot)}$ and $\mathsf{H}'.\mathsf{Out}(\cdot) = \{0,1\}^{p_{\mathsf{out}}(\cdot)}$, where $p_{\mathsf{in}}, p_{\mathsf{out}}$ are some polynomials such that $p_{\mathsf{in}}(\lambda) \ge p_{\mathsf{out}}(\lambda)$ for all $\lambda \in \mathbb{N}$. Next, [10, 45] showed how to use any TCR-secure compression function H' with fixed input length in order to build another TCR-secure function family H for arbitrary, bounded variable-length inputs, meaning that $\mathsf{H.In}(\lambda) = \bigcup_{i \le p(\lambda)} \{0,1\}^i$ and $\mathsf{H.Out}(\lambda) = \mathsf{H}'.\mathsf{Out}(\lambda)$ for some function $p\colon \mathbb{N} \to \mathbb{N}$ all $\lambda \in \mathbb{N}$.

Puncturable PRFs. Consider game PPRF of Fig. 1 associated to a puncturable function family G and an adversary $\mathcal{G}$, where $\mathsf{G.Out}(\lambda)$ is required to be finite for all $\lambda \in \mathbb{N}$ and $\mathcal{G}$ is required to make exactly one oracle query to CH. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{G},\mathcal{G}}(\lambda) = 2\Pr[\mathrm{PPRF}^{\mathcal{G}}_{\mathsf{G}}(\lambda)] - 1$. Let $\delta\colon \mathbb{N} \to \mathbb{R}$ be any function. We say that G is a δ-PPRF-secure if for every PT adversary $\mathcal{G}$ there exists $\lambda_{\delta,\mathcal{G}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{G},\mathcal{G}}(\lambda) \le \delta(\lambda)$ for all $\lambda \ge \lambda_{\delta,\mathcal{G}}$. We say that G is sub-exponentially PPRF-secure if it is $2^{-(\cdot)^\epsilon}$-PPRF-secure for some $0 < \epsilon < 1$. Puncturable PRFs were concurrently and independently introduced in [17, 39, 19]. They can be built by extending the standard PRF construction of Goldreich, Goldwasser and Micali [31].

Digital signature schemes. A digital signature scheme DS defines PT algorithms $\mathsf{DS.Kg}$, $\mathsf{DS.Sig}$, $\mathsf{DS.Ver}$, where $\mathsf{DS.Ver}$ is deterministic. Associated to DS is a collection of input sets $\mathsf{DS.In}$ and a collection of output sets $\mathsf{DS.Out}$, defining all valid messages and signatures for each of security parameters. Key generation algorithm $\mathsf{DS.Kg}$ takes 1^λ to return a signing key sk and a verification key vk. Signing algorithm $\mathsf{DS.Sig}$ takes $1^\lambda, sk$ and a message $m \in \mathsf{DS.In}(\lambda)$ to return a signature $\sigma \in \mathsf{DS.Out}(\lambda)$. Verification algorithm $\mathsf{DS.Ver}$ takes $1^\lambda, vk, m, \sigma$ to return a decision $d \in \{1, 0\}$

```
Game $\mathrm{DIFF}^{\mathcal{D}}_{\mathsf{S}}(\lambda)$
$(\mathrm{P}_0, \mathrm{P}_1, aux) \leftarrow\!\!\$\; \mathsf{S}(1^\lambda)$
$x \leftarrow\!\!\$\; \mathcal{D}(1^\lambda, \mathrm{P}_0, \mathrm{P}_1, aux)$
Return $(\mathrm{P}_0(x) \neq \mathrm{P}_1(x))$
```

```
Game $\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf},\mathsf{S}}(\lambda)$
$b \leftarrow\!\!\$\; \{0,1\}$
$(\mathrm{P}_0, \mathrm{P}_1, aux) \leftarrow\!\!\$\; \mathsf{S}(1^\lambda)$
$\overline{\mathrm{P}} \leftarrow\!\!\$\; \mathsf{Obf}(1^\lambda, \mathrm{P}_b)$
$b' \leftarrow\!\!\$\; \mathcal{O}(1^\lambda, \overline{\mathrm{P}}, aux)$
Return $(b = b')$
```

Figure 2: Games defining difference-security of program sampler S and iO-security of program obfuscator Obf relative to program sampler S.

regarding whether σ is a valid signature of m under vk, where 1 is returned if σ is a valid and 0 otherwise. The correctness condition requires that $\mathsf{DS.Ver}(1^\lambda, vk, m, \sigma) = 1$ for all $\lambda \in \mathbb{N}$, all $(sk, vk) \in [\mathsf{DS.Kg}(1^\lambda)]$, all $m \in \mathsf{DS.In}(\lambda)$ and all $\sigma \in [\mathsf{DS.Sig}(1^\lambda, sk, m)]$. We say that a digital signature scheme DS is deterministic if its signing algorithm $\mathsf{DS.Sig}$ is deterministic.

Obfuscators. An obfuscator is a PT algorithm Obf that on input 1^λ and a program P returns a program $\overline{\mathrm{P}}$ of the same type as P such that $\overline{\mathrm{P}} \equiv \mathrm{P}$. We say that Obf is a circuit obfuscator if it obfuscates circuits, and we say that Obf is a TM obfuscator if it obfuscates TMs. Note that according to our definition of functionally equivalent programs, obfuscation is not defined for TMs that do not halt on some inputs. The polynomial slowdown condition requires that for every TM obfuscator Obf there is a polynomial $p\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for every TM M that halts on all inputs and for every input $x \in \{0,1\}^*$, we have $\mathsf{time}(\overline{\mathrm{M}}, x) \le p(\lambda, \mathsf{time}(\mathrm{M}, x))$ for all $\lambda \in \mathbb{N}$ and all $\overline{\mathrm{M}} \in [\mathsf{Obf}(1^\lambda, \mathrm{M})]$. An analogous slowdown condition trivially holds for any PT circuit obfuscator.

In this work, we discuss indistinguishabilty obfuscation (iO) and differing-inputs obfuscation (diO). The study of these obfuscation notions was initiated in [7]. Later [26, 44] showed how to build and use the former, whereas [18, 1] provided results on the latter. We extend the definitional framework of [12] that uses classes of program samplers to capture different variants of security notions for iO and diO. Specifically, our definitions allow for a unified treatment of polynomial and sub-exponential security of both circuit and TM obfuscation.

Program samplers. A circuit sampler is a PT algorithm $\mathsf{S}^{\mathsf{circ}}$ that on input 1^λ returns a triple $(\mathrm{C}_0, \mathrm{C}_1, aux)$, where $\mathrm{C}_0, \mathrm{C}_1$ are circuits of the same size, number of inputs and number of outputs, and aux is a string. A TM sampler is a PT algorithm S^{tm} that on input 1^λ returns a triple $(\mathrm{M}_0, \mathrm{M}_1, aux)$, where $\mathrm{M}_0, \mathrm{M}_1$ are TMs of the same size, and aux is a string. We require that $\mathrm{M}_0(x)$ and $\mathrm{M}_1(x)$ halt for all $\lambda \in \mathbb{N}$, all $(\mathrm{M}_0, \mathrm{M}_1, aux) \in [\mathsf{S}^{\mathsf{tm}}(1^\lambda)]$ and all $x \in \{0,1\}^*$. We say that S is a program sampler if it is either a circuit sampler or a TM sampler.

Classes of program samplers. We say that a program sampler S produces functionally equivalent programs if $\Pr\left[\, \mathrm{P}_0 \equiv \mathrm{P}_1 \;:\; (\mathrm{P}_0, \mathrm{P}_1, aux) \leftarrow\!\!\$\; \mathsf{S}(1^\lambda) \,\right] = 1$ for all $\lambda \in \mathbb{N}$. Let $\boldsymbol{S}^{\mathsf{circ}}_{\mathrm{eq}}$ be the class of all circuit samplers that produce functionally equivalent circuits, and let $\boldsymbol{S}^{\mathsf{tm}}_{\mathrm{eq}}$ be the class of all TM samplers that produce functionally equivalent TMs. Consider game DIFF of Fig. 2 associated to a program sampler S and an adversary $\mathcal{D}$. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{diff}}_{\mathsf{S},\mathcal{D}}(\lambda) = \Pr[\mathrm{DIFF}^{\mathcal{D}}_{\mathsf{S}}(\lambda)]$. Let $\delta\colon \mathbb{N} \to \mathbb{R}$ be any function. We say that S is δ-DIFF-secure if for every PT adversary $\mathcal{D}$ there exists $\lambda_{\delta,\mathcal{D}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{diff}}_{\mathsf{S},\mathcal{D}}(\lambda) \le \delta(\lambda)$ for all $\lambda \ge \lambda_{\delta,\mathcal{D}}$. We say that S is sub-exponentially DIFF-secure if it is $2^{-(\cdot)^\epsilon}$-DIFF-secure for some $0 < \epsilon < 1$. Let $\boldsymbol{S}^{\mathsf{circ}}_{\delta\text{-diff}}$ be the class of all δ-DIFF-secure circuit samplers, and let $\boldsymbol{S}^{\mathsf{tm}}_{\delta\text{-diff}}$ be the class of all δ-DIFF-secure TM samplers. Informally, difference-security of a program sampler S means that given its output $(\mathrm{P}_0, \mathrm{P}_1, aux)$, it is hard to find an input on which the programs P_0 and P_1 differ.

Indistinguishability obfuscation and differing-inputs obfuscation. Consider game IO of Fig. 2 associated to an obfuscator Obf, a program sampler S and an adversary $\mathcal{O}$. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf},\mathsf{S},\mathcal{O}}(\lambda) = 2\Pr[\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf},\mathsf{S}}(\lambda)] - 1$. Let $\delta\colon \mathbb{N} \to \mathbb{R}$ be any function. Let $\boldsymbol{S}$ be a class of program samplers. We say that Obf is δ-$\boldsymbol{S}$-secure if for every program sampler $\mathsf{S} \in \boldsymbol{S}$ and for every PT adversary $\mathcal{O}$ there exists $\lambda_{\delta,\mathsf{S},\mathcal{O}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf},\mathsf{S},\mathcal{O}}(\lambda) \leq \delta(\lambda)$ for all $\lambda \geq \lambda_{\delta,\mathsf{S},\mathcal{O}}$. We say that Obf is sub-exponentially $\boldsymbol{S}$-secure if it is $2^{-(\cdot)^{\epsilon}}$-$\boldsymbol{S}$-secure for some $0 < \epsilon < 1$.

We say that Obf is a sub-exponentially secure indistinguishability obfuscator for TMs (resp. circuits) if there exists $0 < \epsilon < 1$ such that Obf is $2^{-(\cdot)^{\epsilon}}$–$\boldsymbol{S}^{\mathsf{tm}}_{\mathsf{eq}}$-secure (resp. $2^{-(\cdot)^{\epsilon}}$–$\boldsymbol{S}^{\mathsf{circ}}_{\mathsf{eq}}$-secure). We say that Obf is a differing-inputs obfuscator for TMs (resp. circuits) if for every negligible function $\gamma\colon \mathbb{N} \to \mathbb{R}$ there exists a negligible function $\nu\colon \mathbb{N} \to \mathbb{R}$ such that Obf is ν-$\boldsymbol{S}^{\mathsf{tm}}_{\gamma\text{-diff}}$-secure (resp. ν-$\boldsymbol{S}^{\mathsf{circ}}_{\gamma\text{-diff}}$-secure). Note that ν-$\boldsymbol{S}^{\mathsf{tm}}_{\gamma\text{-diff}}$-security may be unachievable if there exists an infinite number of security parameters $\lambda \in \mathbb{N}$ such that $\gamma(\lambda) > \nu(\lambda)$. We say that Obf is a sub-exponentially secure differing-inputs obfuscator for TMs (resp. circuits) if for every $0 < \epsilon_0 < 1$ and $\gamma = 2^{-(\cdot)^{\epsilon_0}}$ there exists $0 < \epsilon_1 < 1$ such that Obf is $2^{-(\cdot)^{\epsilon_1}}$-$\boldsymbol{S}^{\mathsf{tm}}_{\gamma\text{-diff}}$-secure (resp. $2^{-(\cdot)^{\epsilon_1}}$-$\boldsymbol{S}^{\mathsf{circ}}_{\gamma\text{-diff}}$-secure).

Note that according to our definitions, a sub-exponentially secure differing-inputs obfuscator is not necessarily a polynomially-secure differing-inputs obfuscator. Namely, the former guarantees no security with respect to δ-DIFF-secure program samplers when δ is negligible but not sub-exponentially small. This observation can be used to strengthen our definition of sub-exponentially secure diO. We chose to use the weaker definition, which is simpler to define and which makes our impossibility results stronger.

3 Consistent puncturable digital signature schemes

We start by defining *consistent puncturable digital signature schemes* that will be used for our impossibility results in Section 4. Our construction follows Sahai-Waters signatures [44], and we prove its security assuming OWF and iO.

Informally, a puncturable digital signature scheme allows to 'puncture' its signing key sk at an arbitrary message m^*. The resulting punctured secret key sk^*, punctured at m^*, allows to produce signatures for all messages except for m^*. The puncturability property is similar to the one of puncturable PRFs. We say that a puncturable digital signature scheme is *consistent* if its secret signing key sk and every possible punctured signing key sk^*, that can be derived from sk, deterministically produce the same signatures for all messages except for the punctured message.

We now define a security notion, informally, requiring that no PT adversary should be able to forge a valid signature for the punctured message. The natural formalization of this security notion requires *selective* unforgeability, meaning that an adversary has to choose a message m^* at which the original signing key sk should be punctured. Having received the corresponding pair of punctured signing key sk^* and verification key vk, the goal of the adversary is to produce a valid signature for m^* with respect to the verification key.

Puncturable digital signature schemes. A *puncturable* digital signature scheme DS specifies (beyond the algorithms associated to digital signatures schemes) additional PT algorithms $\mathsf{DS.PKg}$, $\mathsf{DS.PSig}$, where $\mathsf{DS.PSig}$ is deterministic. Punctured key generation algorithm $\mathsf{DS.PKg}$ takes 1^{λ}, a signing key $sk \in [\mathsf{DS.Kg}(1^{\lambda})]$ and a message $m^* \in \mathsf{DS.In}(\lambda)$ to return a "punctured" signing key sk^*. Punctured signing algorithm $\mathsf{DS.PSig}$ takes $1^{\lambda}, sk^*$ and a message $m \in \mathsf{DS.In}(\lambda)$ to return a signature $\sigma \in \mathsf{DS.Out}(\lambda)$. We say that puncturable digital signature scheme DS is *consistent* if $\mathsf{DS.Sig}(1^{\lambda}, sk, m) = \mathsf{DS.PSig}(1^{\lambda}, sk^*, m)$ for all $\lambda \in \mathbb{N}$, all $(sk, vk) \in [\mathsf{DS.Kg}(1^{\lambda})]$, all $m^* \in \mathsf{DS.In}(\lambda)$, all $sk^* \in [\mathsf{DS.PKg}(1^{\lambda}, sk, m^*)]$ and all $m \in \mathsf{DS.In}(\lambda) \setminus \{m^*\}$. Note that DS can be consistent only

```
Game $\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(\lambda)$
$(m^*, st) \leftarrow\$ \mathcal{U}_1(1^\lambda)$
$(sk, vk) \leftarrow\$ \mathsf{DS.Kg}(1^\lambda)$
$sk^* \leftarrow\$ \mathsf{DS.PKg}(1^\lambda, sk, m^*)$
$\sigma^* \leftarrow\$ \mathcal{U}_2(1^\lambda, st, vk, sk^*)$
$d \leftarrow \mathsf{DS.Ver}(1^\lambda, vk, m^*, \sigma^*)$
Return $(d = 1)$
```

Figure 3: Game defining selective unforgeability of puncturable digital signature scheme DS under chosen message attack.

if it is deterministic. More precisely, both $\mathsf{DS.Sig}$ and $\mathsf{DS.PSig}$ should be deterministic. However, determinism is a necessary but not a sufficient condition.

Punctured selective unforgeability under chosen message attack. Consider game PSUFCMA of Fig. 3 associated to a puncturable digital signature scheme DS and an adversary $\mathcal{U}$. For $\lambda \in \mathbb{N}$ let $\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(\lambda) = \Pr[\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(\lambda)]$. Let $\delta \colon \mathbb{N} \to \mathbb{R}$ be any function. We say that DS is δ-PSUFCMA-secure if for every PT adversary $\mathcal{U}$ there exists $\lambda_{\delta,\mathcal{U}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(\lambda) \leq \delta(\lambda)$ for all $\lambda \geq \lambda_{\delta,\mathcal{U}}$. We say that DS is sub-exponentially PSUFCMA-secure if it is $2^{-(\cdot)^\epsilon}$-PSUFCMA-secure for some $0 < \epsilon < 1$.

Our construction. We build a consistent puncturable digital signature scheme DS from a PPRF G, an indistinguishability obfuscator Obf and a OWF F. Our main observation is that a PPRF key gk can be used as a secret key for DS. In order to obtain a punctured key for DS, we puncture gk accordingly. The correctness condition of puncturable PRFs guarantees that DS is consistent. We build a verification key by obfuscating a circuit that embeds the PPRF key gk and a OWF key fk. The circuit takes a message-signature pair (m, σ) and returns 1 if $\mathsf{F.Ev}(1^\lambda, fk, \sigma) = \mathsf{F.Ev}(1^\lambda, fk, \mathsf{G.Ev}(1^\lambda, gk, m))$; it returns 0 otherwise.

Puncturable digital signature scheme PUNC-DS. Let $s \colon \mathbb{N} \to \mathbb{N}$ be a polynomial. Let G be a puncturable function family. Let F be a function family such that $\mathsf{F.In} = \mathsf{G.Out}$. Let Obf be a circuit obfuscator. We build a consistent puncturable digital signature scheme $\mathsf{DS} = \mathsf{PUNC\text{-}DS}[\mathsf{G}, \mathsf{F}, \mathsf{Obf}, s]$ as follows. Let $\mathsf{DS.In}(\lambda) = \mathsf{G.In}(\lambda)$ and $\mathsf{DS.Out}(\lambda) = \mathsf{G.Out}(\lambda)$ for all $\lambda \in \mathbb{N}$, and

```
Algorithm $\mathsf{DS.Kg}(1^\lambda)$
$gk \leftarrow\$ \mathsf{G.Kg}(1^\lambda)$ ;  $fk \leftarrow\$ \mathsf{F.Kg}(1^\lambda)$
$\overline{\mathrm{C}} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{Pad}_{s(\lambda)}(\mathrm{C}_{1^\lambda,gk,fk}))$
Return $(gk, \overline{\mathrm{C}})$

Circuit $\mathrm{C}_{1^\lambda,gk,fk}(m, \sigma)$
$\sigma' \leftarrow \mathsf{G.Ev}(1^\lambda, gk, m)$
$y' \leftarrow \mathsf{F.Ev}(1^\lambda, fk, \sigma')$
If $(y' = \mathsf{F.Ev}(1^\lambda, fk, \sigma))$ then return 1
Else return 0
```

```
Algorithm $\mathsf{DS.PKg}(1^\lambda, gk, m^*)$
Return $\mathsf{G.PKg}(1^\lambda, gk, m^*)$

Algorithm $\mathsf{DS.Ver}(1^\lambda, \overline{\mathrm{C}}, m, \sigma)$
Return $\overline{\mathrm{C}}(m, \sigma)$

Algorithm $\mathsf{DS.Sig}(1^\lambda, gk, m)$
Return $\mathsf{G.Ev}(1^\lambda, gk, m)$

Algorithm $\mathsf{DS.PSig}(1^\lambda, gk^*, m)$
Return $\mathsf{G.PEv}(1^\lambda, gk^*, m)$
```

We say that DS is *well-defined* if $s(\lambda) \geq |\mathrm{C}_{1^\lambda,gk,fk}|$ for all $\lambda \in \mathbb{N}$, all $gk \in [\mathsf{G.Kg}(1^\lambda)]$ and all $fk \in [\mathsf{F.Kg}(1^\lambda)]$.

The following says that a PSUFCMA-secure, consistent punctured digital signature scheme can be built assuming OWF and iO.

Theorem 3.1 *Let* G *be a sub-exponentially* PPRF*-secure function family such that* $\mathsf{G.In}(\lambda), \mathsf{G.Out}(\lambda) \subseteq \bigcup_{i \le p_0(\lambda)} \{0,1\}^i$ *for some polynomial* p_0 *and all* $\lambda \in \mathbb{N}$. *Let* F *be a sub-exponentially* OW*-secure function family such that* $\mathsf{F.In} = \mathsf{G.Out}$ *and* $\mathsf{F.Out}(\lambda) \subseteq \bigcup_{i \le p_1(\lambda)} \{0,1\}^i$ *for some polynomial* p_1 *and all* $\lambda \in \mathbb{N}$. *Let* Obf *be a sub-exponentially* $\boldsymbol{S}_{\mathsf{eq}}^{\mathsf{circ}}$*-secure circuit obfuscator. Let* $\mathsf{DS} = \mathsf{PUNC\text{-}DS}[\mathsf{G}, \mathsf{F}, \mathsf{Obf}, s]$. *Then (1)* DS *is well-defined, and (2)* DS *is sub-exponentially* PSUFCMA*-secure.*

In order to prove that DS is PSUFCMA-secure, we show that an adversary can not find the value of $\mathsf{G.Ev}(1^\lambda, gk, m^*)$ for a challenge message m^*, even given the obfuscated verification-key circuit that contains gk. In the proof, we puncture gk at m^* to get a punctured key gk^*, and construct a functionally equivalent verification-key circuit that embeds gk^* along with $y^* = \mathsf{F.Ev}(1^\lambda, \mathit{fk}, \mathsf{G.Ev}(1^\lambda, gk, m^*))$. The new verification key accepts σ as a valid signature for m^* if and only if $y^* = \mathsf{F.Ev}(1^\lambda, \mathit{fk}, \sigma)$, whereas the verification of signatures for all other messages $m \neq m^*$ remains the same. First, we use the iO-security of Obf to switch the verification circuits. Then we use the PPRF-security of G, followed by the OWF-security of F to show that no adversary can find the value of $\mathsf{G.Ev}(1^\lambda, gk, m^*)$ from gk^* and y^*.

Proof of Theorem 3.1: For any $\lambda \in \mathbb{N}$ let $s(\lambda)$ be a polynomial upper bound on $\max(|\mathrm{C}^1_{1^\lambda, gk, \mathit{fk}}|\,,\, |\mathrm{C}^2_{1^\lambda, gk^*, \mathit{fk}, m^*, y^*}|)$ where the circuits are defined in Fig. 4 and the maximum is over all $\mathit{fk} \in [\mathsf{F.Kg}(1^\lambda)]$, $gk \in [\mathsf{G.Kg}(1^\lambda)]$, $m^* \in \mathsf{G.In}(\lambda)$, $gk^* \in [\mathsf{G.PKg}(1^\lambda, gk, m^*)]$ and $y^* \in \mathsf{F.Out}(\lambda)$. This implies part (1) of the theorem, meaning that DS is well-defined.

Let $0 < \epsilon_{\mathsf{pprf}} < 1$ be a constant for which G is $2^{-(\cdot)^{\epsilon_{\mathsf{pprf}}}}$–PPRF-secure. Let $0 < \epsilon_{\mathsf{ow}} < 1$ be a constant for which F is $2^{-(\cdot)^{\epsilon_{\mathsf{ow}}}}$–OW-secure. Let $0 < \epsilon_{\mathsf{io}} < 1$ be a constant for which Obf is $2^{-(\cdot)^{\epsilon_{\mathsf{io}}}}$–$\boldsymbol{S}_{\mathsf{eq}}^{\mathsf{circ}}$-secure. Let $\epsilon = \frac{1}{2}\min(\epsilon_{\mathsf{pprf}}, \epsilon_{\mathsf{ow}}, \epsilon_{\mathsf{io}})$. We now prove claim (2) by showing that DS is $2^{-(\cdot)^{\epsilon}}$-PSUFCMA-secure.

Let $\mathcal{U}$ be a PT adversary. Consider the games and associated circuits of Fig. 4. Lines not annotated with comments are common to all games. Game $\mathrm{G}_0(\lambda)$ is equivalent to $\mathrm{PSUFCMA}_{\mathsf{DS}}^{\mathcal{U}}(\lambda)$, so for all $\lambda \in \mathbb{N}$ we have

$$\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)]. \tag{1}$$

The proof proceeds in three steps. In the first step we transition from game G_0 to game G_1, by replacing circuit $\mathrm{C}^1_{1^\lambda, gk, \mathit{fk}}$ with circuit $\mathrm{C}^2_{1^\lambda, gk^*, \mathit{fk}, m^*, y^*}$. On input (m, σ) circuit $\mathrm{C}^1_{1^\lambda, gk, \mathit{fk}}$ compares $\mathsf{F.Ev}(1^\lambda, \mathit{fk}, \sigma)$ with $\mathsf{F.Ev}(1^\lambda, \mathit{fk}, \mathsf{G.Ev}(1^\lambda, gk, m))$, where gk is a PPRF key. In contrast, circuit $\mathrm{C}^2_{1^\lambda, gk^*, \mathit{fk}, m^*, y^*}$ contains the corresponding punctured key gk^*, punctured at the challenge message m^*. In order to process inputs that contain message m^*, it uses an embedded value $y^* = \mathsf{F.Ev}(1^\lambda, \mathit{fk}, \mathsf{G.Ev}(1^\lambda, gk, m^*))$ instead. As a result, the circuits are functionally equivalent. We build a circuit sampler $\mathsf{S} \in \boldsymbol{S}_{\mathsf{eq}}^{\mathsf{circ}}$ and a PT adversary $\mathcal{O}$ against the sub-exponential iO-security of Obf relative to S such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] = \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf},\mathsf{S},\mathcal{O}}(\lambda). \tag{2}$$

```
Circuit Sampler $\mathsf{S}(1^\lambda)$
$(m^*, st) \leftarrow_{\$} \mathcal{U}_1(1^\lambda)$ ; $\mathit{fk} \leftarrow_{\$} \mathsf{F.Kg}(1^\lambda)$
$gk \leftarrow_{\$} \mathsf{G.Kg}(1^\lambda)$ ; $gk^* \leftarrow_{\$} \mathsf{G.PKg}(1^\lambda, gk, m^*)$
$r^* \leftarrow \mathsf{G.Ev}(1^\lambda, gk, m^*)$ ; $y^* \leftarrow \mathsf{F.Ev}(1^\lambda, \mathit{fk}, r^*)$
$\mathrm{C}_1 \leftarrow \mathsf{Pad}_{s(\lambda)}(\mathrm{C}^1_{1^\lambda, gk, \mathit{fk}})$ ; $\mathrm{C}_0 \leftarrow \mathsf{Pad}_{s(\lambda)}(\mathrm{C}^2_{1^\lambda, gk^*, \mathit{fk}, m^*, y^*})$
$aux \leftarrow (st, gk^*, m^*)$ ; Return $(\mathrm{C}_0, \mathrm{C}_1, aux)$
```

```
Adversary $\mathcal{O}(1^\lambda, \overline{\mathrm{C}}, aux)$
$(st, gk^*, m^*) \leftarrow aux$
$\sigma^* \leftarrow_{\$} \mathcal{U}_2(1^\lambda, st, \overline{\mathrm{C}}, gk^*)$
$b \leftarrow \overline{\mathrm{C}}(m^*, \sigma^*)$
Return $b$
```

```
Games $\mathrm{G}_0(\lambda)$–$\mathrm{G}_2(\lambda)$
$(m^*, st) \leftarrow\!\!\$\, \mathcal{U}_1(1^\lambda)$ ; $fk \leftarrow\!\!\$\, \mathsf{F.Kg}(1^\lambda)$ ; $gk \leftarrow\!\!\$\, \mathsf{G.Kg}(1^\lambda)$ ; $gk^* \leftarrow\!\!\$\, \mathsf{G.PKg}(1^\lambda, gk, m^*)$
  $r^* \leftarrow \mathsf{G.Ev}(1^\lambda, gk, m^*)$ ; $y^* \leftarrow \mathsf{F.Ev}(1^\lambda, fk, r^*)$ ; $\mathrm{C}_{\mathsf{ver}} \leftarrow \mathrm{C}^1_{1^\lambda, gk, fk}$   // $\mathrm{G}_0$
  $r^* \leftarrow \mathsf{G.Ev}(1^\lambda, gk, m^*)$ ; $y^* \leftarrow \mathsf{F.Ev}(1^\lambda, fk, r^*)$ ; $\mathrm{C}_{\mathsf{ver}} \leftarrow \mathrm{C}^2_{1^\lambda, gk^*, fk, m^*, y^*}$   // $\mathrm{G}_1$
  $r^* \leftarrow\!\!\$\, \mathsf{G.Out}(\lambda)$ ;   $y^* \leftarrow \mathsf{F.Ev}(1^\lambda, fk, r^*)$ ; $\mathrm{C}_{\mathsf{ver}} \leftarrow \mathrm{C}^2_{1^\lambda, gk^*, fk, m^*, y^*}$   // $\mathrm{G}_2$
$\overline{\mathrm{C}} \leftarrow\!\!\$\, \mathsf{Obf}(1^\lambda, \mathsf{Pad}_{s(\lambda)}(\mathrm{C}_{\mathsf{ver}}))$ ; $\sigma^* \leftarrow\!\!\$\, \mathcal{U}_2(1^\lambda, st, \overline{\mathrm{C}}, gk^*)$ ; $b \leftarrow \overline{\mathrm{C}}(m^*, \sigma^*)$ ; Return $(b = 1)$
```

```
Circuit $\mathrm{C}^1_{1^\lambda, gk, fk}(m, \sigma)$
$\sigma' \leftarrow \mathsf{G.Ev}(1^\lambda, gk, m)$
$y' \leftarrow \mathsf{F.Ev}(1^\lambda, fk, \sigma')$
If $(y' = \mathsf{F.Ev}(1^\lambda, fk, \sigma))$ then return 1
Else return 0
```

```
Circuit $\mathrm{C}^2_{1^\lambda, gk^*, fk, m^*, y^*}(m, \sigma)$
If $(m \neq m^*)$ then
  $\sigma' \leftarrow \mathsf{G.PEv}(1^\lambda, gk^*, m)$
  $y' \leftarrow \mathsf{F.Ev}(1^\lambda, fk, \sigma')$
Else $y' \leftarrow y^*$
If $(y' = \mathsf{F.Ev}(1^\lambda, fk, \sigma))$ then return 1
Else return 0
```

Figure 4: **Games for proof of Theorem 3.1.**

Next, in the transition from game G_1 to game G_2 we use the PPRF-security of G in order to replace r^* by a uniformly random value. We build a PT adversary $\mathcal{G}$ against the PPRF-security of G such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] = \mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{G},\mathcal{G}}(\lambda). \tag{3}$$

```
Adversary $\mathcal{G}^{\mathrm{CH}}(1^\lambda)$
$(m^*, st) \leftarrow\!\!\$\, \mathcal{U}_1(1^\lambda)$ ; $fk \leftarrow\!\!\$\, \mathsf{F.Kg}(1^\lambda)$ ; $(gk^*, r^*) \leftarrow \mathrm{CH}(m^*)$
$y^* \leftarrow \mathsf{F.Ev}(1^\lambda, fk, r^*)$ ; $\overline{\mathrm{C}} \leftarrow\!\!\$\, \mathsf{Obf}(1^\lambda, \mathsf{Pad}_{s(\lambda)}(\mathrm{C}^2_{1^\lambda, gk^*, fk, m^*, y^*}))$
$\sigma^* \leftarrow\!\!\$\, \mathcal{U}_2(1^\lambda, st, \overline{\mathrm{C}}, gk^*)$ ; $b \leftarrow \overline{\mathrm{C}}(m^*, \sigma^*)$ ; Return $b$
```

Finally, in order to win game G_2, adversary $\mathcal{U}$ has to find a preimage of y^* under the one-way function F with key fk. We build a PT adversary $\mathcal{F}$ against the OW-security of F such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{G}_2(\lambda)] = \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathcal{F}}(\lambda). \tag{4}$$

```
Adversary $\mathcal{F}(1^\lambda, fk, y^*)$
$(m^*, st) \leftarrow\!\!\$\, \mathcal{U}_1(1^\lambda)$ ; $gk \leftarrow\!\!\$\, \mathsf{G.Kg}(1^\lambda)$ ; $gk^* \leftarrow\!\!\$\, \mathsf{G.PKg}(1^\lambda, gk, m^*)$
$\overline{\mathrm{C}} \leftarrow\!\!\$\, \mathsf{Obf}(1^\lambda, \mathsf{Pad}_{s(\lambda)}(\mathrm{C}^2_{1^\lambda, gk^*, fk, m^*, y^*}))$
$\sigma^* \leftarrow\!\!\$\, \mathcal{U}_2(1^\lambda, st, \overline{\mathrm{C}}, gk^*)$ ; Return $\sigma^*$
```

Let $\lambda_{\mathsf{S},\mathcal{O}} \in \mathbb{N}$ be such that $\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf},\mathsf{S},\mathcal{O}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{io}}}}$ for all $\lambda \geq \lambda_{\mathsf{S},\mathcal{O}}$. Let $\lambda_{\mathcal{G}} \in \mathbb{N}$ be such that $\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{G},\mathcal{G}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{pprf}}}}$ for all $\lambda \geq \lambda_{\mathcal{G}}$. Let $\lambda_{\mathcal{F}} \in \mathbb{N}$ be such that $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathcal{F}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{ow}}}}$ for all $\lambda \geq \lambda_{\mathcal{F}}$. Then there exists $\lambda_{\mathcal{U}} \in \mathbb{N}$ such that the following holds for all $\lambda \geq \lambda_{\mathcal{U}}$:

$$\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(\lambda) = \sum_{i=0}^{1}(\Pr[\mathrm{G}_i(\lambda)] - \Pr[\mathrm{G}_{i+1}(\lambda)]) + \Pr[\mathrm{G}_2(\lambda)] \tag{5}$$

$$= \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf},\mathsf{S},\mathcal{O}}(\lambda) + \mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{G},\mathcal{G}}(\lambda) + \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathcal{F}}(\lambda) \tag{6}$$

$$
\begin{aligned}
&\leq 2^{-\lambda^{\epsilon_{\mathsf{io}}}} + 2^{-\lambda^{\epsilon_{\mathsf{pprf}}}} + 2^{-\lambda^{\epsilon_{\mathsf{ow}}}} && (7)\\
&\leq 3 \cdot 2^{-\lambda^{2\epsilon}} && (8)\\
&\leq 3 \cdot 2^{-(2\lambda)^{\epsilon}} = 2^{\log_2 3 - (2\lambda)^{\epsilon}} && (9)\\
&\leq 2^{-\lambda^{\epsilon}} && (10)
\end{aligned}
$$

Equation (5) follows from Equation (1) for all $\lambda \in \mathbb{N}$. Equation (6) follows from Equations (2)–(4) for all $\lambda \in \mathbb{N}$. Equation (7) holds for all $\lambda \geq \max(\lambda_{\mathsf{S},\mathcal{O}}, \lambda_{\mathcal{G}}, \lambda_{\mathcal{F}})$, according to the sub-exponential security of Obf, G and F. Equation (8) follows from our choice of ϵ, namely because $2\epsilon \leq \epsilon_{\mathsf{io}}$, $2\epsilon \leq \epsilon_{\mathsf{pprf}}$ and $2\epsilon \leq \epsilon_{\mathsf{ow}}$. Equation (9) holds for all $\lambda \in \mathbb{N}$ such that $\lambda^{2\epsilon} \geq (2\lambda)^{\epsilon}$, requiring that $\lambda \geq 2$. Equation (10) holds whenever $\log_2 3 - 2^{\epsilon}\lambda^{\epsilon} \leq -\lambda^{\epsilon}$, requiring that $\lambda \geq \left(\frac{\log_2 3}{2^{\epsilon}-1}\right)^{1/\epsilon}$. Therefore, it suffices to set

$$
\lambda_{\mathcal{U}} = \max\left(\lambda_{\mathsf{S},\mathcal{O}}, \lambda_{\mathcal{G}}, \lambda_{\mathcal{F}}, 2, \left\lceil \left(\frac{\log_2 3}{2^{\epsilon}-1}\right)^{1/\epsilon} \right\rceil \right).
$$

This completes the proof. ▌

4 Impossibility of differing-inputs obfuscation for TMs

In this section we show that differing-inputs obfuscation for Turing Machines is impossible. In order to disprove sub-exponentially secure diO for TMs, we assume only the existence of sub-exponentially secure one-way functions. Furthermore, we show that polynomially secure diO for TMs is also impossible, additionally assuming sub-exponentially secure iO.

We construct a sub-exponentially difference-secure TM sampler, meaning that given a pair of TMs produced by this sampler it is hard to find an input on which these TMs produce different outputs. The proof of difference-security is the core part of our work. It requires to carefully specify how to choose parameters for our sampler in a way that does not introduce any circular dependencies. Besides proving difference-security, we also show that there exists an adversary that can distinguish between obfuscations of TMs that are produced by the sampler regardless of the used obfuscator. Together these claims imply the impossibility of diO for TMs.

The blueprint for impossibility results. The first black-box attack on differing-inputs obfuscation was presented by Garg, Gentry, Halevi and Wichs (GGHW) [27]. They introduced a novel *special-purpose obfuscation* assumption and showed that it contradicts diO. Our impossibility result follows the high-level idea from their work, but we achieve it using concrete assumptions. We now explain the core ideas of our impossibility result, which roughly follow GGHW.

We construct a TM sampler S^{tm} that returns TMs $\mathrm{M}^0, \mathrm{M}^1$ along with an auxiliary information string *aux*. The sampler generates a key pair (sk, vk) for a digital signature scheme DS, and its output depends on these keys. TM M^0 returns 0 on every input. TM M^1 returns 1 if and only if it gets a valid message-signature pair as input, corresponding to the verification key vk; it returns 0 otherwise. The auxiliary information string *aux* is an iO-obfuscation of a TM $\mathrm{M}^{\mathsf{aux}}$. The latter embeds the signing key sk and takes a TM $\overline{\mathrm{M}}$ as input, which for our purpose will normally be a diO-obfuscation of M^0 or M^1. $\mathrm{M}^{\mathsf{aux}}$ returns the result of running $\overline{\mathrm{M}}$ on a message-signature pair that is produced using its embedded signing key sk.

In order to determine whether a TM $\overline{\mathrm{M}}$ is an obfuscation of M^0 or M^1, one can run $\mathrm{M}^{\mathsf{aux}}$ with $\overline{\mathrm{M}}$ as input. According to the construction of $\mathrm{M}^{\mathsf{aux}}$, it will return $b \in \{0, 1\}$ if and only if $\overline{\mathrm{M}}$ is an obfuscation of M^b. To prove difference-security of S^{tm}, we will show that it is hard to find a valid message-signature pair given $(\mathrm{M}^0, \mathrm{M}^1, aux)$. The main technical challenge of the proof is to show that aux (the obfuscation of $\mathrm{M}^{\mathsf{aux}}$) properly hides the embedded signing key sk, which does not naturally follow from the security of indistinguishability obfuscation.

Turing Machine sampler TM-SAMP. Let $s_0, \ell, n, t_0, t_1, s_1 \colon \mathbb{N} \to \mathbb{N}$ be polynomials. Let $\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}, \mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ be TM obfuscators. Let H be a function family such that $\mathsf{H.In}(\lambda) = \{0,1\}^*$ and $\mathsf{H.Out}(\lambda) \subseteq \bigcup_{i \le p_0(\lambda)} \{0,1\}^i$ for some polynomial p_0 and all $\lambda \in \mathbb{N}$. Let DS be a deterministic digital signature scheme such that $\mathsf{DS.In}(\lambda) = \{0,1\}^{\ell(\lambda)}$ and $\mathsf{DS.Out}(\lambda) \subseteq \bigcup_{i \le p_1(\lambda)} \{0,1\}^i$ for some polynomial p_1 and all $\lambda \in \mathbb{N}$. We build a TM sampler $\mathsf{S}^{\mathsf{tm}} = \mathsf{TM\text{-}SAMP}\ [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}, \mathsf{H}, \mathsf{DS}, \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}, s_0, \ell, n, t_0, t_1, s_1]$ as follows:

```
TM Sampler $\mathsf{S}^{\mathsf{tm}}(1^\lambda)$
$(sk, vk) \leftarrow_{\$} \mathsf{DS.Kg}(1^{n(\lambda)})$
$hk \leftarrow_{\$} \mathsf{H.Kg}(1^\lambda)$
$h \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk})$
$\mathrm{M}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$
$\mathrm{M}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^\lambda, hk, h})$
$\mathrm{M}_{\mathsf{aux}} \leftarrow \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk})$
$aux \leftarrow_{\$} \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}(1^{n(\lambda)}, \mathrm{M}_{\mathsf{aux}})$
Return $(\mathrm{M}_0, \mathrm{M}_1, aux)$
```

```
TM $\mathrm{M}^0(\mathrm{M}, 1^t, m, \sigma)$
Return 0

TM $\mathrm{M}^1_{1^\lambda, hk, h}(\mathrm{M}, 1^t, m, \sigma)$
$h' \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M})$
If $(h' \ne h)$ then return 0
Return $\mathsf{UTM}^t_{\mathrm{M}}(m, \sigma)$
```

```
TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}(\overline{\mathrm{M}})$
If $(|\overline{\mathrm{M}}| \ne \ell(\lambda))$ then return 0
$\sigma \leftarrow \mathsf{DS.Sig}(1^{n(\lambda)}, sk, \langle \overline{\mathrm{M}} \rangle_{\ell(n(\lambda))})$
$d \leftarrow \mathsf{UTM}^{t_1(\lambda)}_{\overline{\mathrm{M}}}(\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma)$
Return $d$

TM $\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}(m, \sigma)$
If $(|m| \ne \ell(\lambda))$ then return 0
$d \leftarrow \mathsf{DS.Ver}(1^{n(\lambda)}, vk, \langle m \rangle_{\ell(n(\lambda))}, \sigma)$
Return $d$
```

We say that S^{tm} is *well-defined* if $s_0(\lambda) \ge |\mathrm{M}^0|$, $s_0(\lambda) \ge |\mathrm{M}^1_{1^\lambda, hk, h}|$, $\ell(n(\lambda)) \ge \ell(\lambda)$, $t_0(\lambda) \ge \mathsf{time}(\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}, (m, \sigma))$, $t_1(\lambda) \ge \mathsf{time}(\overline{\mathrm{M}}, (\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma))$ and $s_1(\lambda) \ge |\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}|$ for all $\lambda \in \mathbb{N}$, all $hk \in [\mathsf{H.Kg}(1^\lambda)]$, all $h \in \mathsf{H.Out}(\lambda)$, all $\mathrm{M} \in \{\mathrm{M}^0, \mathrm{M}^1_{1^\lambda, hk, h}\}$, all $\overline{\mathrm{M}} \in [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}(1^\lambda, \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}))]$, all $(sk, vk) \in [\mathsf{DS.Kg}(1^{n(\lambda)})]$, all $m \in \{0,1\}^{\ell(\lambda)}$ and all $\sigma \in \mathsf{DS.Out}(n(\lambda))$.

Core design ideas behind TM-SAMP. Note that TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}$ takes as input an obfuscated TM $\overline{\mathrm{M}}$ and computes the signature σ for message $\langle \overline{\mathrm{M}} \rangle_{\ell(n(\lambda))}$, where the latter denotes $\overline{\mathrm{M}}$ padded to size $\ell(n(\lambda))$. It then uses a Universal Turing Machine UTM to simulate $\overline{\mathrm{M}}$ on input x for the duration of $t_1(\lambda)$ steps, where $x = (\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma)$. The idea of computing a signature on a message that depends on $\overline{\mathrm{M}}$ was already proposed in GGHW [27], with the goal of avoding a trivial attack against the difference-security of the sampler. Specifically, if a fixed message-signature pair $(m_{\mathsf{ch}}, \sigma_{\mathsf{ch}})$ was used for all inputs of $\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}$, then a difference-security adversary could construct a sequence of TMs that each reveals a single bit of $(m_{\mathsf{ch}}, \sigma_{\mathsf{ch}})$ when used as an input $\overline{\mathrm{M}}$ to $\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}$. This would allow adversary to recover the message-signature pair bit-by-bit.

Turing Machine $\mathrm{M}^1_{1^\lambda, hk, h}$ takes an input $x = (\mathrm{M}, 1^t, m, \sigma)$, where M is a TM, 1^t is the unary representation of some integer $t \in \mathbb{N}$, and (m, σ) is a message-signature pair. We use a target collision-resistant function family H in order to ensure that $\mathrm{M}^1_{1^\lambda, hk, h}$ can return 1 only if $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}$. This is achieved by embeding a key hk for H and the value $h = \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk})$ into $\mathrm{M}^1_{1^\lambda, hk, h}$, and by returning 0 whenever $h \ne \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M})$. If $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}$ is satisfied, then $\mathrm{M}^1_{1^\lambda, hk, h}$ uses a Universal Turing Machine UTM to simulate M on input (m, σ) for the duration of t steps. TM $\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}$ is designed to return 1 if and only if its input $x = (m, \sigma)$ is a valid message-signature pair with respect to a verification key vk for the digital signature scheme DS. Our impossibility results require the choice of DS to depend on the construction of $\mathrm{M}^1_{1^\lambda, hk, h}$, so embedding vk directly into the latter would have introded a circular dependency between the two. Instead we have to resort to the above approach of embedding vk into a separate TM.

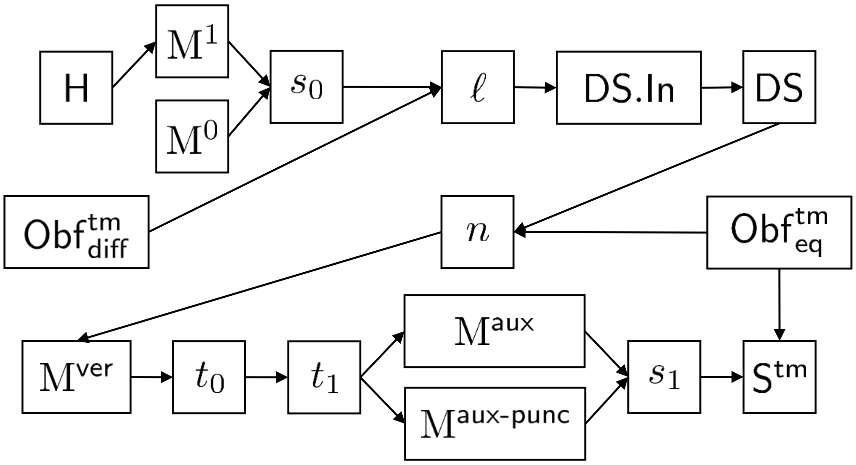
H
M^1
M^0
s_0
ℓ
DS.In
DS
$\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{diff}}$
n
$\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}$
M^{ver}
t_0
t_1
M^{aux}
$M^{\mathsf{aux\text{-}punc}}$
s_1
S^{tm}

Figure 5: **Parameter dependencies in TM-SAMP for the proof of Theorem 4.1.**

According to our definitions, two TMs can be functionally equivalent only if both of them halt on all inputs. The notion of functional equivalence is further used for the definitions of program samplers and obfuscation. This means that whenever a TM needs to simulate the code of another TM, it is required to use a Universal Turing Machine UTM and specify the number of steps for the simulation. Otherwise, the simulated TMs would not be guaranteed to halt.

Parameters of TM-SAMP. Fig. 5 shows the dependencies between all schemes and parameters that will be used to instantiate the construction of TM-SAMP in Theorem 4.1. Let us introduce the notation that is used in this picture. For any two entities A and B, an arrow from A to B means that the construction, or the choice, of B depends on A. The relations are transitive, meaning that we do not draw a direct arrow from A to B in the case if B is already reachable from A. TM $M^{\mathsf{aux\text{-}punc}}$ will be used only for the proof of security and is defined in Fig. 6.

The construction of TM-SAMP is parameterized by polynomials s_0, s_1, t_0, t_1, ℓ and n. Polynomials s_0, s_1 denote the size to which some of our TMs must be padded prior to obfuscating them. This stems from our definition of program samplers that are required to return programs of the same size. Polynomials t_0, t_1 are used to indicate the number of steps that must be done when simulating various TMs using a Universal Turing Machine UTM. Our definition of a well-defined instantiation of TM-SAMP specifies lower bounds for t_0, t_1 that ensure the correctness of the attack that we will design against the sub-exponential (d)iO-security of $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ with respect to S^{tm}. Polynomial ℓ will be defined to upper-bound the size of any obfuscation $\overline{M}$ of programs M^0 and $M^1_{1^\lambda,hk,h}$, when obfuscator $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ is used. Note that $M^{\mathsf{aux}}_{1^\lambda,sk,vk}$ rejects all inputs $\overline{M}$ of size different than $\ell(\lambda)$; our attack will pad all obfuscations of M^0 and $M^1_{1^\lambda,hk,h}$ to size $\ell(\lambda)$, using the padding operator $\mathsf{Pad}_{\ell(\lambda)}(\cdot)$ that is assumed to produce functionally equivalent TMs as per Section 2. Polynomial n is used to set security parameters for schemes DS and $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$. Specifically, if the TM sampler S^{tm} is instantiated with a security parameter $\lambda \in \mathbb{N}$, then its construction uses these two schemes, each with the security parameter $n(\lambda)$.

In order for our proof of difference-security to work, if a $2^{-(\cdot)^\epsilon}$-security is assumed for either of DS or $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$, then the choice of polynomial n will depend on ϵ. This leads to an inconvenient dependency: DS uses $n(\lambda)$ as its security parameter, but the choice of polynomial n depends on the choice of DS. Ideally, we would have liked to choose a digital signature scheme DS such that $\mathsf{DS.Out}(n(\lambda)) = \{0,1\}^{\ell(\lambda)}$, because DS is used to sign messages that are TMs of size $\ell(\lambda)$. However, since we do not know n ahead of choosing DS, we require that for all $\lambda \in \mathbb{N}$ we have $\mathsf{DS.Out}(\lambda) = \{0,1\}^{\ell(\lambda)}$ and $\ell(n(\lambda)) \geq \ell(\lambda)$, resulting in $\mathsf{DS.Out}(n(\lambda)) = \{0,1\}^{\ell(n(\lambda))}$. We then use an injective string padding to map TMs (i.e. their string representations) of length $\ell(\lambda)$ into strings of length $\ell(n(\lambda))$. The injectivity of padding is necessary for the proof of difference-security of S^{tm}. In order to ensure that the requirement $\ell(n(\lambda)) \geq \ell(\lambda)$ is satisfied, we will choose polynomials ℓ, n such that $\ell(\lambda+1) \geq \ell(\lambda)$ and $n(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$.

Limitations and extensions. Our definition of TM samplers in Section 2 requires them to return TMs that halt on all inputs. One could argue that this definition is still insufficient for the purpose of obfuscation. Namely, a sampler can produce TMs that have significantly different running times, and it might not be reasonable to expect an obfuscator to properly hide the difference in the running times. We note that this does not hinder our results because we can artificially alter our TMs M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$ to have the same running times, by adding void instructions to the definition of M^0.

The construction of $\mathsf{TM\text{-}SAMP}$ uses a TM obfuscator $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$ that in our theorem statements will be assumed to be sub-exponentially $\boldsymbol{S}^{\mathsf{tm}}_{\mathrm{eq}}$-secure. It is used to produce auxiliary information by obfuscating TMs $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$ and $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$. We use a TM obfuscator for readability, but we note that a sub-exponentially $\boldsymbol{S}^{\mathsf{circ}}_{\mathrm{eq}}$-secure *circuit obfuscator* could be used instead. There are no circular dependencies preventing us from redefining these two TMs as circuits.

According to Fig. 5, the size of $\mathrm{M}^{\mathsf{aux}}$ depends on the maximum size of TMs M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$, and in particular it might be larger than these TMs. This means that our impossibility result might not hold if we restrict our attention to TM samplers whose auxiliary information strings *aux* are required to be shorter than the size of the corresponding TMs M^0 and M^1. GGHW [27] circumvent this limitation in their impossibility result by using a CRHF to compute and then sign a hash of the TM that is passed inside their auxiliary-information program, rather than signing the TM itself. Our proof techniques do not seem to be compatible with such approach.

Impossibility results. We now formally state our results. Theorem 4.1 shows how to choose parameters for $\mathsf{TM\text{-}SAMP}$ such that the resulting TM sampler is simultaneously well-defined and difference-secure. Theorem 4.2 shows that any well-defined instantiation of $\mathsf{TM\text{-}SAMP}$ produces TMs that can not be securely obfuscated.

Theorem 4.1 *Let $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ be a TM obfuscator. Let H be a sub-exponentially TCR-secure function family such that $\mathsf{H.In}(\lambda) = \{0,1\}^*$ and $\mathsf{H.Out}(\lambda) \subseteq \bigcup_{i \leq p_0(\lambda)} \{0,1\}^i$ for some polynomial p_0 and all $\lambda \in \mathbb{N}$. Then there are polynomials $s_0, \ell \colon \mathbb{N} \to \mathbb{N}$ such that the following is true. Let DS be a sub-exponentially PSUFCMA-secure, consistent puncturable digital signature scheme such that $\mathsf{DS.In}(\lambda) = \{0,1\}^{\ell(\lambda)}$ and $\mathsf{DS.Out}(\lambda) \subseteq \bigcup_{i \leq p_1(\lambda)} \{0,1\}^i$ for some polynomial p_1 and all $\lambda \in \mathbb{N}$. Let $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$ be a sub-exponentially $\boldsymbol{S}^{\mathsf{tm}}_{\mathrm{eq}}$-secure TM obfuscator. Then there are polynomials $n, t_0, t_1, s_1 \colon \mathbb{N} \to \mathbb{N}$ such that the following is true. Let $\mathsf{S}^{\mathsf{tm}} = \mathsf{TM\text{-}SAMP}\ [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}},\ \mathsf{H},\ \mathsf{DS},\ \mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}, s_0, \ell, n, t_0, t_1, s_1]$. Then (1) S^{tm} is well-defined, and (2) S^{tm} is sub-exponentially DIFF-secure.*

We defer the proof of Theorem 4.1 until after we show how to use this theorem to state and prove our main claims regarding the impossibility of differing-inputs obfuscation for TMs.

Theorem 4.2 *Let $s_0, \ell, n, t_0, t_1, s_1 \colon \mathbb{N} \to \mathbb{N}$ be polynomials. Let $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}, \mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ be TM obfuscators. Let H be a function family with $\mathsf{H.In}(\lambda) = \{0,1\}^*$ and $\mathsf{H.Out}(\lambda) \subseteq \bigcup_{i \leq p_0(\lambda)} \{0,1\}^i$ for some polynomial p_0 and all $\lambda \in \mathbb{N}$. Let DS be a deterministic digital signature scheme such that $\mathsf{DS.In}(\lambda) = \{0,1\}^{\ell(\lambda)}$ and $\mathsf{DS.Out}(\lambda) \subseteq \bigcup_{i \leq p_1(\lambda)} \{0,1\}^i$ for some polynomial p_1 and all $\lambda \in \mathbb{N}$. Let $\mathsf{S}^{\mathsf{tm}} = \mathsf{TM\text{-}SAMP}\ [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}, \mathsf{H}, \mathsf{DS}, \mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}, s_0, \ell, n, t_0, t_1, s_1]$. Assume that S^{tm} is well-defined. Then there exists a PT adversary $\mathcal{O}$ such that $\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}},\mathsf{S}^{\mathsf{tm}},\mathcal{O}}(\lambda) = 1$.*

Proof of Theorem 4.2: We build a PT adversary $\mathcal{O}$ against the (d)iO-security of $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ relative to S^{tm} as follows:

```
Adversary $\mathcal{O}(1^\lambda, \overline{\mathrm{M}}, aux)$
$\overline{\mathrm{M}}_{\mathsf{aux}} \leftarrow aux$
$b' \leftarrow \overline{\mathrm{M}}_{\mathsf{aux}}(\mathsf{Pad}_{\ell(\lambda)}(\overline{\mathrm{M}}))$
Return $b'$
```

Adversary $\mathcal{O}$ takes $1^\lambda, \overline{\mathrm{M}}, aux$ as input, where $\overline{\mathrm{M}}$ is an obfuscation of either TM M^0 or TM $\mathrm{M}^1_{1^\lambda,hk,h}$ that was produced by the obfuscator $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}$ in game $\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}},\mathsf{S}^{\mathsf{tm}}}(\lambda)$, and aux is an auxiliary information string. The goal of $\mathcal{O}$ is to guess which of M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$ was obfuscated. It should return 0 if $\overline{\mathrm{M}}$ is an obfuscation of M^0, and it should return 1 otherwise.

Adversary $\mathcal{O}$ parses auxiliary information string aux into a TM $\overline{\mathrm{M}}_{\mathsf{aux}}$. The latter is an obfuscation of TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$, which was computed in S^{tm} using obfuscator $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$. Next, $\mathcal{O}$ pads $\overline{\mathrm{M}}$ to construct a functionally equivalent TM of size $\ell(\lambda)$ and passes it as input to $\overline{\mathrm{M}}_{\mathsf{aux}}$. According to the construction of $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$, the latter returns 1 if and only if $\overline{\mathrm{M}}$ is an obfuscation of TM $\mathrm{M}^1_{1^\lambda,hk,h}$. Adversary $\mathcal{O}$ returns the same value to win the game. This concludes the proof of Theorem 4.2. ▌

Next, Theorem 4.3 shows the impossibility of a polynomially secure diO, whereas Theorem 4.4 shows the impossibility of a sub-exponentially secure diO.

Theorem 4.3 *Let* Obf *be a Turing Machine obfuscator. Assume the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation for Turing Machines. Then* Obf *is not a differing-inputs obfuscator.*

We now prove Theorem 4.3. Let $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$ be a sub-exponentially $\boldsymbol{S}^{\mathsf{tm}}_{\mathrm{eq}}$-secure TM obfuscator. Theorem 3.1 shows how to build a sub-exponentially PSUFCMA-secure, consistent puncturable digital signature scheme DS assuming only sub-exponentially secure OWF and sub-exponentially secure iO. For a moment, assume that we can build a TCR-secure function family H with $\mathsf{H.In}(\lambda) = \{0,1\}^*$ for all $\lambda \in \mathbb{N}$ just from sub-exponentially secure OWFs (which is not known to be true, and we address this below). Then according to Theorem 4.1, we can build a TM sampler S^{tm} that is (1) well-defined and (2) sub-exponentially DIFF-secure. But Theorem 4.2 shows that there exists an efficient adversary that breaks the IO-security of Obf with respect to S^{tm}. Therefore, Obf is not a differing-inputs obfuscator.

In order to build a TCR-secure function family H from a sub-exponentially secure OWF, the statements of Theorem 4.1 and Theorem 4.2 can be relaxed to require $\mathsf{H.In}(\lambda) = \{0,1\}^{2^\lambda}$ for all $\lambda \in \mathbb{N}$. This change will still ensure the correctness of S^{tm}, which requires that H can process inputs of length $|\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}|$. The size of $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ in our construction is bounded polynomially in the security parameter. But the reason we have to use a hash function that can process inputs of arbitrary, super-polynomially bounded lengths is because the size of $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ is not known prior to fixing H (as shown in Fig. 5).

As noted in Section 2, Shoup [45] shows how to build a TCR-secure function family H for arbitrary, bounded variable-length inputs from any TCR-secure compression function family with fixed input size. The latter is shown to be achievable from OWFs by Rompel [43]. We note that the key size of Shoup's construction grows logarithmically with the maximum input length of the constructed function family, which is still polynomiallly bounded in the case of H that was proposed above. Furthermore, the super-polynomial bound on the message lengths does not introduce any difficulties for the security reduction of Shoup's construction. This is because the loss of security during the reduction depends on the length of the messages that are chosen by a PT adversary, rather than by the (super-polynomial) bound on the messages supported by the scheme.

This concludes the proof of Theorem 4.3. Note that we ruled out the existence of polynomially-secure differing-inputs obfuscation even with respect to *sub-exponentially* secure TM samplers, which is a stronger version of difference-security than the one required by our definition of polynomially-secure differing-inputs obfuscation.

Theorem 4.4 *Let* Obf *be a Turing Machine obfuscator. Assume the existence of sub-exponentially secure one-way functions. Then* Obf *is not a sub-exponentially secure differing-inputs obfuscator.*

To prove Theorem 4.4, assume for a contradiction that Obf is a sub-exponentially secure differing-inputs obfuscator. According to our definitions, it implies the existence of sub-exponentially secure indistinguishability obfuscation. The rest of the proof is identical to the proof of Theorem 4.3. It results in constructing a sub-exponentially difference-secure TM sampler S^{tm} that can not be securely obfuscated by Obf. Thus, we get a contradiction.

Finally, we now prove Theorem 4.1.

Proof of Theorem 4.1: We start by proving part (1) of the theorem. Specifically, we choose polynomials $s_0, \ell, n, t_0, t_1, s_1 \colon \mathbb{N} \to \mathbb{N}$ such that S^{tm} is well-defined.

We now specify polynomials $s_0, \ell \colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$ let $s_0(\lambda)$ be a polynomial upper bound on $\max(|\mathrm{M}^0|, |\mathrm{M}^1_{1^\lambda, hk, h}|)$ where the maximum is over all $hk \in [\mathsf{H.Kg}(1^\lambda)]$ and $h \in \mathsf{H.Out}(\lambda)$. For any $\lambda \in \mathbb{N}$ let $\ell(\lambda)$ be a polynomial upper bound on $\max(|\overline{\mathrm{M}}|)$ such that $\ell(\lambda) \leq \ell(\lambda{+}1)$, where the maximum is over all $hk \in [\mathsf{H.Kg}(1^\lambda)]$, $h \in \mathsf{H.Out}(\lambda)$, $\mathrm{M} \in \{\mathrm{M}^0, \mathrm{M}^1_{1^\lambda, hk, h}\}$ and $\overline{\mathrm{M}} \in [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}(1^\lambda, \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}))]$. Note that the requirement that $\ell(\lambda) \leq \ell(\lambda + 1)$ for all $\lambda \in \mathbb{N}$ is trivially achievable by removing all terms with negative coefficients from the polynomial.

We now specify a constant $0 < \epsilon < 1$ this is required to define polynomial n. Let $0 < \epsilon_{\mathsf{tcr}} < 1$ be a constant such that H is $2^{-(\cdot)^{\epsilon_{\mathsf{tcr}}}}$-TCR-secure. Let $0 < \epsilon_{\mathsf{psuf}} < 1$ be a constant such that DS is $2^{-(\cdot)^{\epsilon_{\mathsf{psuf}}}}$-PSUFCMA-secure. Let $0 < \epsilon_{\mathsf{io}} < 1$ be a constant such that $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$ is $2^{-(\cdot)^{\epsilon_{\mathsf{io}}}}$-$\boldsymbol{S}^{\mathsf{tm}}_{\mathrm{eq}}$-secure. Let $\epsilon = \min(\frac{1}{2}\epsilon_{\mathsf{tcr}}, \epsilon_{\mathsf{psuf}}, \epsilon_{\mathsf{io}})$. Later we will prove that S^{tm} is $2^{-(\cdot)^{\epsilon}}$-DIFF-secure.

We now specify polynomial $n \colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$ let $n(\lambda) = (2\lambda + \ell(\lambda) + 3)^{\lceil 1/\epsilon \rceil}$. Note that for any $\lambda \in \mathbb{N}$ we have $n(\lambda) \geq \lambda$, and earlier we required that $\ell(\lambda + 1) \geq \ell(\lambda)$ for all $\lambda \in \mathbb{N}$. It follows that $\ell(n(\lambda)) \geq \ell(\lambda)$ for all $\lambda \in \mathbb{N}$, as required for S^{tm} to be well-defined. Let Inv_n be a deterministic, PT algorithm that takes $1^{\lambda'}$ to return the smallest $\lambda \in \mathbb{N}$ such that $n(\lambda) \geq \lambda'$. We note that $\mathsf{Inv}_n(1^{n(\lambda)}) = \lambda$ for all $\lambda \in \mathbb{N}$ since n is injective, which follows from the requirement that $\ell(\lambda + 1) \geq \ell(\lambda)$ for all $\lambda \in \mathbb{N}$.

We now specify polynomials $n, t_0, t_1, \ell_1 \colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$ let $t_0(\lambda)$ be a polynomial upper bound on the maximum running time of $\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}(m, \sigma)$ where the maximum is over all $(sk, vk) \in [\mathsf{DS.Kg}(1^{n(\lambda)})]$, $m \in \{0,1\}^{\ell(\lambda)}$ and $\sigma \in \mathsf{DS.Out}(n(\lambda))$. For any $\lambda \in \mathbb{N}$ let $t_1(\lambda)$ be a polynomial upper bound on the maximum running time of $\overline{\mathrm{M}}(\mathrm{M}^{\mathsf{ver}}_{1^\lambda, vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma)$ where the maximum is over all $hk \in [\mathsf{H.Kg}(1^\lambda)]$, $h \in \mathsf{H.Out}(\lambda)$, $\mathrm{M} \in \{\mathrm{M}^0, \mathrm{M}^1_{1^\lambda, hk, h}\}$, $\overline{\mathrm{M}} \in [\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{diff}}(1^\lambda, \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}))]$, $(sk, vk) \in [\mathsf{DS.Kg}(1^{n(\lambda)})]$ and $\sigma \in \mathsf{DS.Out}(n(\lambda))$. For any $\lambda \in \mathbb{N}$ let $s_1(\lambda)$ be a polynomial upper bound on $\max(|\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}|, |\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda, sk^*, vk, m', b}|)$ where the maximum is over all $(sk, vk) \in [\mathsf{DS.Kg}(1^{n(\lambda)})]$, $m' \in \{0,1\}^{\ell(\lambda)}$, $sk^* \in [\mathsf{DS.PKg}(1^{n(\lambda)}, sk, \langle m' \rangle_{\ell(n(\lambda))})]$ and $b \in \{0,1\}$.

We proceed to prove part (2) of Theorem 4.1, namely that S^{tm} is $2^{-(\cdot)^{\epsilon}}$-DIFF-secure. The main challenge of the proof is to show that the signing key sk of DS can not be extracted from $\mathrm{M}^{\mathsf{aux}}_{1^\lambda, sk, vk}$,

```
Games $\mathrm{G}_0(\lambda)$–$\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)$
$(sk, vk) \leftarrow_{\$} \mathsf{DS.Kg}(1^{n(\lambda)})$
$hk \leftarrow_{\$} \mathsf{H.Kg}(1^\lambda)$
$h \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
$\mathrm{M}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$
$\mathrm{M}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^\lambda,hk,h})$
$\mathrm{M}_{\mathsf{aux}} \leftarrow \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk})$
$aux \leftarrow_{\$} \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}(1^{n(\lambda)}, \mathrm{M}_{\mathsf{aux}})$
$x \leftarrow_{\$} \mathcal{D}(1^\lambda, \mathrm{M}_0, \mathrm{M}_1, aux)$
$(\mathrm{M}, 1^t, m, \sigma) \leftarrow x$
$d_0 \leftarrow (\mathrm{M}_0(x) \neq \mathrm{M}_1(x))$
$d_1 \leftarrow (\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
res $\leftarrow d_0$                                  // $\mathrm{G}_0$
res $\leftarrow (d_0 \wedge d_1 \wedge m \geq 0)$          // $\mathrm{G}_{1,0}$
...
res $\leftarrow (d_0 \wedge d_1 \wedge m \geq 2^{\ell(\lambda)})$  // $\mathrm{G}_{1,2^{\ell(\lambda)}}$
Return res
```

```
Games $\mathrm{G}_{1,i}(\lambda)$–$\mathrm{G}_{1,i+1}(\lambda)$
$(sk, vk) \leftarrow_{\$} \mathsf{DS.Kg}(1^{n(\lambda)})$
$hk \leftarrow_{\$} \mathsf{H.Kg}(1^\lambda)$
$h \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
$\mathrm{M}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$
$\mathrm{M}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^\lambda,hk,h})$
$m' \leftarrow \langle i \rangle_{\ell(\lambda)}$ ; $b \leftarrow \mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}(m')$
$m^* \leftarrow \langle m' \rangle_{\ell(n(\lambda))}$
$sk^* \leftarrow_{\$} \mathsf{DS.PKg}(1^{n(\lambda)}, sk, m^*)$
$\mathrm{M}_{\mathsf{tmp}} \leftarrow \mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$ ;              $z \leftarrow i$      // $\mathrm{G}_{1,i}$
$\mathrm{M}_{\mathsf{tmp}} \leftarrow \mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ ;  $z \leftarrow i$      // $\mathrm{G}_{1,i,A}$
$\mathrm{M}_{\mathsf{tmp}} \leftarrow \mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ ;  $z \leftarrow i+1$  // $\mathrm{G}_{1,i,B}$
$\mathrm{M}_{\mathsf{tmp}} \leftarrow \mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$ ;              $z \leftarrow i+1$  // $\mathrm{G}_{1,i+1}$
$aux \leftarrow_{\$} \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}(1^{n(\lambda)}, \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}_{\mathsf{tmp}}))$
$x \leftarrow_{\$} \mathcal{D}(1^\lambda, \mathrm{M}_0, \mathrm{M}_1, aux)$
$(\mathrm{M}, 1^t, m, \sigma) \leftarrow x$
$d_0 \leftarrow (\mathrm{M}_0(x) \neq \mathrm{M}_1(x))$
$d_1 \leftarrow (\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
Return $(d_0 \wedge d_1 \wedge m \geq z)$
```

```
TM $\mathrm{M}^0(\mathrm{M}, 1^t, m, \sigma)$
Return 0

TM $\mathrm{M}^1_{1^\lambda,hk,h}(\mathrm{M}, 1^t, m, \sigma)$
$h' \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M})$
If $(h' \neq h)$ then return 0
Return $\mathsf{UTM}^t_{\mathrm{M}}(m, \sigma)$

TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}(\overline{\mathrm{M}})$
If $(|\overline{\mathrm{M}}| \neq \ell(\lambda))$ then return 0
$\sigma \leftarrow \mathsf{DS.Sig}(1^{n(\lambda)}, sk, \langle \overline{\mathrm{M}} \rangle_{\ell(n(\lambda))})$
$d \leftarrow \mathsf{UTM}^{t_1(\lambda)}_{\overline{\mathrm{M}}}(\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma)$
Return $d$
```

```
TM $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}(m, \sigma)$
If $(|m| \neq \ell(\lambda))$ then return 0
Return $\mathsf{DS.Ver}(1^{n(\lambda)}, vk, \langle m \rangle_{\ell(n(\lambda))}, \sigma)$

TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}(\overline{\mathrm{M}})$
If $(|\overline{\mathrm{M}}| \neq \ell(\lambda))$ then return 0
If $(\overline{\mathrm{M}} = m')$ then return $b$
$\sigma \leftarrow \mathsf{DS.PSig}(1^{n(\lambda)}, sk^*, \langle \overline{\mathrm{M}} \rangle_{\ell(n(\lambda))})$
$d \leftarrow \mathsf{UTM}^{t_1(\lambda)}_{\overline{\mathrm{M}}}(\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}, 1^{t_0(\lambda)}, \overline{\mathrm{M}}, \sigma)$
Retrn $d$
```

Figure 6: **Games for proof of Theorem 4.1.**

meaning that the $\boldsymbol{S}^{\mathsf{tm}}_{\mathsf{eq}}$-secure obfuscation is sufficient to hide sk. In our proof this is implicit. The core idea of the proof is to consider the exponential number of messages from $\mathsf{DS.In}(n(\lambda))$ and for each of them we argue that a PT adversary is unlikely to produce a signature for this message. This implies that it is hard to find an input on which TMs M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$ return different outputs.

Let $\mathcal{D}$ be a PT adversary. Consider the games and associated TMs of Fig. 6. Lines not annotated with comments are common to all games. Game $\mathrm{G}_0(\lambda)$ is equivalent to $\mathrm{DIFF}^{\mathcal{D}}_{\mathsf{S}^{\mathsf{tm}}}(\lambda)$, so for all $\lambda \in \mathbb{N}$ we have

$$\mathsf{Adv}^{\mathsf{diff}}_{\mathsf{S}^{\mathsf{tm}},\mathcal{D}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)]. \tag{11}$$

Let us discuss the transitions between hybrid games that will be used in our proof. Let $\lambda \in \mathbb{N}$. In order to transition from game $\mathrm{G}_0(\lambda)$ to game $\mathrm{G}_{1,0}(\lambda)$ we claim that if adversary $\mathcal{D}$ wins in game $\mathrm{DIFF}^{\mathcal{D}}_{\mathsf{S}^{\mathsf{tm}}}(\lambda)$ then it must return a differing-input $x = (\mathrm{M}, 1^t, m, \sigma)$ such that $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$. Otherwise, one could use this adversary in order to break the TCR-security of H. Next, we consider an exponential number of games, going from game $\mathrm{G}_{1,0}(\lambda)$ to game $\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)$. Each game corresponds to a unique value of message m that can be taken as an input by TM $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$. For any $i \in \{0,1,\ldots,2^{\ell(\lambda)}\}$, adversary $\mathcal{D}$ wins in game $\mathrm{G}_{1,i}(\lambda)$ if and only if it returns $x = (\mathrm{M}, 1^t, m, \sigma)$ such that $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$, $m \geq i$ and $\mathrm{M}_0(x) \neq \mathrm{M}_1(x)$. According to this definition, it is impossible to win game $\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)$ because TM $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ rejects whenever it takes a message m as input such that $|m| \neq \ell(\lambda)$ (whereas the length of m in this game is required to be at least $\ell(\lambda)+1$). We now need to show that for each $i \in \{0,1,\ldots,2^{\ell(\lambda)}-1\}$ the success probabilities of adversary $\mathcal{D}$ in games $\mathrm{G}_{1,i}(\lambda)$ and $\mathrm{G}_{1,i+1}(\lambda)$ are sub-exponentially close.

Let $i \in \{0,1,\ldots,2^{\ell(\lambda)}-1\}$. We split the transition from game $\mathrm{G}_{1,i}(\lambda)$ to game $\mathrm{G}_{1,i+1}(\lambda)$ into three steps. Specifically, we consider a sequence of games $\mathrm{G}_{1,i}(\lambda)$, $\mathrm{G}_{1,i,A}(\lambda)$, $\mathrm{G}_{1,i,B}(\lambda)$ and $\mathrm{G}_{1,i+1}(\lambda)$. Games $\mathrm{G}_{1,i,A}(\lambda)$ and $\mathrm{G}_{1,i,B}(\lambda)$ generate aux as an obfuscation of TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ instead of an obfuscation of TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$, where $m' = i$ and the used obfuscator is $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$. As opposed to TM $\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}$, note that TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ contains a punctured signing key sk^* for DS that is punctured at the message $m^* = \langle m' \rangle_{\ell(n(\lambda))}$. Both TMs are defined to produce the same outputs on all inputs $\overline{\mathrm{M}} \neq m'$, which is achieved because the punctured digital signature scheme DS is assumed to be consistent. (Recall that the latter requires that sk and sk^* return the same signatures for all messages except m^*.) Furthermore, TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ is hardwired to return $b = \mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}(m')$ on input $\overline{\mathrm{M}} = m'$, meaning that the TMs are functionally equivalent. We use it to claim that the success probabilities of adversary $\mathcal{D}$ (1) in games $\mathrm{G}_{1,i}(\lambda)$ and $\mathrm{G}_{1,i,A}(\lambda)$, and (2) in games $\mathrm{G}_{1,i,B}(\lambda)$ and $\mathrm{G}_{1,i+1}(\lambda)$ – are sub-exponentially close. Namely, if $\mathcal{D}$ can distinguish between any pair of these games with a better than sub-exponentially small probability, then one can use $\mathcal{D}$ to break the iO-security of obfuscator $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$.

It remains to discuss the transition from game $\mathrm{G}_{1,i,A}(\lambda)$ to game $\mathrm{G}_{1,i,B}(\lambda)$. The difference between these games is that the former requires $m \geq i$ as a part of its winning condition, whereas the later requires $m \geq i+1$. Both of these games set aux as an obfuscation of TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$, where sk^* is punctured at $m^* = \langle m' \rangle_{\ell(n(\lambda))}$ and $m' = i$. Note that adversary $\mathcal{D}$ can only have a different success probability in both games if it is capable of forging a signature on message m^* given any information it might be able to extract from TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$. However, $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$ does not contain any information that could help to forge the signature of message m^* (only bit b depends on the challenge signature, but $\mathcal{D}$ can attempt to guess it). Therefore, we can use the PSUFCMA-security of DS to bound the difference in adversary's success probability when transitioning between games $\mathrm{G}_{1,i,A}(\lambda)$ and $\mathrm{G}_{1,i,B}(\lambda)$.

Below we will prove the following claims:
Claim 1. There exists a PT adversary $\mathcal{H}$ against the TCR-security of H such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_{1,0}(\lambda)] \leq \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda). \tag{12}$$

Claim 2. There exist TM samplers $\mathsf{S}^{\mathsf{tm}}_0, \mathsf{S}^{\mathsf{tm}}_1$ and a PT adversary $\mathcal{O}$ against the IO-security of $\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}}$ relative to $\mathsf{S}^{\mathsf{tm}}_0$ and $\mathsf{S}^{\mathsf{tm}}_1$, such that for all $\lambda \in \mathbb{N}$ we have

$$\sum_{i=0}^{2^{\ell(\lambda)}-1} \left(\Pr[\mathrm{G}_{1,i}(\lambda)] - \Pr[\mathrm{G}_{1,i,A}(\lambda)]\right) \leq 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}^{\mathsf{tm}}_0,\mathcal{O}}(n(\lambda)), \tag{13}$$

$$\sum_{i=0}^{2^{\ell(\lambda)}-1} (\Pr[\mathrm{G}_{1,i,B}(\lambda)] - \Pr[\mathrm{G}_{1,i+1}(\lambda)]) \leq 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}^{\mathsf{tm}}_1,\mathcal{O}}(n(\lambda)). \tag{14}$$

Claim 3. There exists a PT adversary $\mathcal{U}$ against the PSUFCMA-security of DS such that for all $\lambda \in \mathbb{N}$ we have

$$\sum_{i=0}^{2^{\ell(\lambda)}-1} (\Pr[\,\mathrm{G}_{1,i,A}(\lambda)\,] - \Pr[\,\mathrm{G}_{1,i,B}(\lambda)\,]) \leq 2^{\ell(\lambda)+1} \cdot \mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(n(\lambda)). \tag{15}$$

Finally, we claim that no adversary can win against $\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)$. Let $x = (\mathrm{M}, 1^t, m, \sigma)$ be the output of adversary $\mathcal{D}$ in game $\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)$. Adversary $\mathcal{D}$ wins the game if the following three conditions are simultaneously true: $\mathrm{M}^0(x) \neq \mathrm{M}^1_{1^\lambda,hk,h}(x)$, $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ and $|m| > \ell(\lambda)$. The first condition requires $\mathrm{M}^1_{1^\lambda,hk,h}(x)$ to return 1. The second condition means that $\mathrm{M}^1_{1^\lambda,hk,h}(x)$ will return the output of $\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}(m,\sigma)$. However, according to the third condition, the latter returns 0. Therefore,

$$\Pr[\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)] = 0. \tag{16}$$

We now show that there exists $\lambda_{\mathcal{D}} \in \mathbb{N}$ such that for all $\lambda \geq \lambda_{\mathcal{D}}$ we have $\mathsf{Adv}^{\mathsf{diff}}_{\mathsf{S}^{\mathsf{tm}},\mathcal{D}}(\lambda) \leq 2^{-\lambda^\epsilon}$. By definition, this means that S^{tm} is $2^{-(\cdot)^\epsilon}$-DIFF-secure.

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{diff}}_{\mathsf{S}^{\mathsf{tm}},\mathcal{D}}(\lambda) &= (\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_{1,0}(\lambda)]) \\
&\quad + \sum_{i=0}^{2^{\ell(\lambda)}-1} (\Pr[\mathrm{G}_{1,i}(\lambda)] - \Pr[\mathrm{G}_{1,i+1}(\lambda)]) + \Pr[\mathrm{G}_{1,2^{\ell(\lambda)}}(\lambda)] && (17) \\
&\leq \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda) + 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}^{\mathsf{tm}}_0,\mathcal{O}}(n(\lambda)) \\
&\quad + 2^{\ell(\lambda)+1} \cdot \mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(n(\lambda)) + 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}^{\mathsf{tm}}_1,\mathcal{O}}(n(\lambda)) && (18) \\
&\leq 2^{-\lambda^{\epsilon_{\mathsf{tcr}}}} + 2^{\ell(\lambda)} \cdot \left(2^{-n(\lambda)^{\epsilon_{\mathsf{io}}}} + 2 \cdot 2^{-n(\lambda)^{\epsilon_{\mathsf{psuf}}}} + 2^{-n(\lambda)^{\epsilon_{\mathsf{io}}}}\right) && (19) \\
&\leq 2^{-\lambda^{\epsilon_{\mathsf{tcr}}}} + 2^{\ell(\lambda)+1} \cdot \left(2^{-n(\lambda)^{\epsilon_{\mathsf{io}}}} + 2^{-n(\lambda)^{\epsilon_{\mathsf{psuf}}}} + 2^{-n(\lambda)^{\epsilon_{\mathsf{io}}}}\right) && (20) \\
&\leq 2^{-\lambda^{2\epsilon}} + 2^{\ell(\lambda)+1} \cdot 3 \cdot 2^{-n(\lambda)^{\epsilon}} && (21) \\
&= 2^{-\lambda^{2\epsilon}} + 2^{\ell(\lambda)+1+\log_2 3-(2\lambda+\ell(\lambda)+3)^{\lceil 1/\epsilon \rceil \cdot \epsilon}} && (22) \\
&\leq 2^{-\lambda^{2\epsilon}} + 2^{-(2\lambda)^{\epsilon}} && (23) \\
&\leq 2^{-(2\lambda)^{\epsilon}} + 2^{-(2\lambda)^{\epsilon}} = 2^{1-(2\lambda)^{\epsilon}} && (24) \\
&\leq 2^{-\lambda^{\epsilon}}. && (25)
\end{aligned}$$

Let $\lambda_{\mathcal{H}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{tcr}}}}$ for all $\lambda \geq \lambda_{\mathcal{H}}$. Let $\lambda_{\mathcal{U}} \in \mathbb{N}$ such that $\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{psuf}}}}$ for all $\lambda \geq \lambda_{\mathcal{U}}$. For $b \in \{0,1\}$ let $\lambda_{\mathsf{S}^{\mathsf{tm}}_b,\mathcal{O}} \in \mathbb{N}$ be such that $\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}^{\mathsf{tm}}_b,\mathcal{O}}(\lambda) \leq 2^{-\lambda^{\epsilon_{\mathsf{io}}}}$ for all $\lambda \geq \lambda_{\mathsf{S}^{\mathsf{tm}}_b,\mathcal{O}}$.

Equation (17) follows from Equation (11) for all $\lambda \in \mathbb{N}$. Equation (18) follows from equations (12)–(16) for all $\lambda \in \mathbb{N}$. Equation (19) holds for all $\lambda \in \mathbb{N}$ such that $\lambda \geq \lambda_{\mathcal{H}}$ and $n(\lambda) \geq \max(\lambda_{\mathsf{S}_0^{\mathsf{tm}},\mathcal{O}}, \lambda_{\mathcal{U}}, \lambda_{\mathsf{S}_1^{\mathsf{tm}},\mathcal{O}})$. Equation (20) holds for all $\lambda \in \mathbb{N}$. Equation (21) is obtained by expanding ϵ according to its definition, namely by using the following relations: $2\epsilon \leq \epsilon_{\mathsf{tcr}}$, $\epsilon \leq \epsilon_{\mathsf{psuf}}$ and $\epsilon \leq \epsilon_{\mathsf{io}}$. Equation (22) is obtained by expanding $n(\lambda)$ according to its definition. Equation (23) holds for all $\lambda \in \mathbb{N}$, because for any polynomial $\ell\colon \mathbb{N} \to \mathbb{N}$, any constant $0 < \epsilon < 1$ and all $\lambda \in \mathbb{N}$ we have

$$\begin{aligned}
&\ell(\lambda) + 1 + \log_2 3 - (2\lambda + \ell(\lambda) + 3)^{\lceil 1/\epsilon \rceil \cdot \epsilon} \\
\leq\, &\ell(\lambda) + 1 + \log_2 3 - (2\lambda + \ell(\lambda) + 3) \\
<\, &-2\lambda \leq -(2\lambda)^{\epsilon}.
\end{aligned}$$

Equation (24) holds for all $\lambda \in \mathbb{N}$ such that $\lambda^{2\epsilon} \geq (2\lambda)^{\epsilon}$, requiring that $\lambda \geq 2$. Equation (25) holds for all $\lambda \in \mathbb{N}$ such that $1 - 2^{\epsilon}\lambda^{\epsilon} \leq -\lambda^{\epsilon}$, requiring that $\lambda \geq \left(\frac{1}{2^{\epsilon}-1}\right)^{1/\epsilon}$. Therefore, it suffices to set

$$\lambda_{\mathcal{D}} = \max\left(\lambda_{\mathcal{H}}, \mathsf{Inv}_n(1^{\lambda_{\mathsf{S}_0^{\mathsf{tm}},\mathcal{O}}}), \mathsf{Inv}_n(1^{\lambda_{\mathcal{U}}}), \mathsf{Inv}_n(1^{\lambda_{\mathsf{S}_1^{\mathsf{tm}},\mathcal{O}}}), 2, \left\lceil (2^{\epsilon}-1)^{-1/\epsilon} \right\rceil\right).$$

This completes the proof. We now prove Claims 1-3.

Proof of Claim 1. We build a PT adversary $\mathcal{H}$ against the TCR-security of H such that for all $\lambda \in \mathbb{N}$ we have $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_{1,0}(\lambda)] \leq \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H},\mathcal{H}}(\lambda)$.

```
Adversary $\mathcal{H}_1(1^\lambda)$
$(sk, vk) \leftarrow\$ \mathsf{DS.Kg}(1^{n(\lambda)})$
$st \leftarrow (sk, vk)$
Return $(\mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}, st)$
```

```
Adversary $\mathcal{H}_2(1^\lambda, st, hk)$
$(sk, vk) \leftarrow st$ ; $h \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
$\mathrm{M}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$ ; $\mathrm{M}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^\lambda,hk,h})$
$aux \leftarrow\$ \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}(1^{n(\lambda)}, \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}))$
$(\mathrm{M}, 1^t, m, \sigma) \leftarrow\$ \mathcal{D}(1^\lambda, \mathrm{M}_0, \mathrm{M}_1, aux)$ ; Return M
```

Let $x = (\mathrm{M}, 1^t, m, \sigma)$ be an output of adversary $\mathcal{D}$ in games $\mathrm{G}_0(\lambda)$ and $\mathrm{G}_{1,0}(\lambda)$ (note that the input distribution of $\mathcal{D}$ is the same in both games). If these games produce different outcomes for the same x, it means that $\mathrm{M}^0(x) \neq \mathrm{M}^1_{1^\lambda,hk,h}(x)$ and $\mathrm{M} \neq \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$. According to the construction of M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$ it follows that $\mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}) = \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M})$. Whenever this happens, adversary $\mathcal{H}$ wins in game $\mathrm{TCR}^{\mathcal{H}}_{\mathsf{H}}(\lambda)$ by returning $x_0 = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ and $x_1 = \mathrm{M}$. This proves the claim.

Proof of Claim 2. We build TM samplers $\mathsf{S}_0^{\mathsf{tm}}, \mathsf{S}_1^{\mathsf{tm}}$ and a PT adversary $\mathcal{O}$ against the IO-security of $\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}$ relative to $\mathsf{S}_0^{\mathsf{tm}}$ and $\mathsf{S}_1^{\mathsf{tm}}$, such that for all $\lambda \in \mathbb{N}$ we have $\sum_{i=0}^{2^{\ell(\lambda)}-1} (\Pr[\mathrm{G}_{1,i}(\lambda)] - \Pr[\mathrm{G}_{1,i,A}(\lambda)]) \leq 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}},\mathsf{S}_0^{\mathsf{tm}},\mathcal{O}}(n(\lambda))$ and $\sum_{i=0}^{2^{\ell(\lambda)}-1} (\Pr[\mathrm{G}_{1,i,B}(\lambda)] - \Pr[\mathrm{G}_{1,i+1}(\lambda)]) \leq 2^{\ell(\lambda)} \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}},\mathsf{S}_1^{\mathsf{tm}},\mathcal{O}}(n(\lambda))$.

Below, on the left we (simultaneously) define the TM samplers $\mathsf{S}_0^{\mathsf{tm}}$ and $\mathsf{S}_1^{\mathsf{tm}}$ that differ at the commented lines and have the uncommented lines in common. On the right, we define the PT adversary $\mathcal{O}$.

```
TM Samplers $\mathsf{S}_0^{\mathsf{tm}}(1^{\lambda'})$, $\mathsf{S}_1^{\mathsf{tm}}(1^{\lambda'})$
$\lambda \leftarrow \mathsf{Inv}_n(1^{\lambda'})$ ; $i \leftarrow\!\$\ \{0,1\}^{\ell(\lambda)}$
$(sk, vk) \leftarrow\!\$\ \mathsf{DS.Kg}(1^{n(\lambda)})$
$hk \leftarrow\!\$\ \mathsf{H.Kg}(1^{\lambda})$ ; $h \leftarrow \mathsf{H.Ev}(1^{\lambda}, hk, \mathrm{M}^{\mathsf{ver}}_{1^{\lambda},vk})$
$\tilde{\mathrm{M}}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$ ; $\tilde{\mathrm{M}}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^{\lambda},hk,h})$
$m' \leftarrow \langle i \rangle_{\ell(\lambda)}$ ; $b \leftarrow \mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}(m')$
$m^* \leftarrow \langle m' \rangle_{\ell(n(\lambda))}$ ; $sk^* \leftarrow\!\$\ \mathsf{DS.PKg}(1^{n(\lambda)}, sk, m^*)$
$\mathrm{M}_{\mathsf{aux}} \leftarrow \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk})$
$\mathrm{M}_{\mathsf{aux\text{-}punc}} \leftarrow \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^{\lambda},sk^*,vk,m',b})$
$\mathrm{M}_1 \leftarrow \mathrm{M}_{\mathsf{aux}}$ ; $\mathrm{M}_0 \leftarrow \mathrm{M}_{\mathsf{aux\text{-}punc}}$ ; $z \leftarrow i$  // $\mathsf{S}_0^{\mathsf{tm}}$
$\mathrm{M}_0 \leftarrow \mathrm{M}_{\mathsf{aux}}$ ; $\mathrm{M}_1 \leftarrow \mathrm{M}_{\mathsf{aux\text{-}punc}}$ ; $z \leftarrow i+1$ // $\mathsf{S}_1^{\mathsf{tm}}$
$aux \leftarrow (\tilde{\mathrm{M}}_0, \tilde{\mathrm{M}}_1, vk, z)$ ; return $(\mathrm{M}_0, \mathrm{M}_1, aux)$
```

```
Adversary $\mathcal{O}(1^{\lambda'}, \overline{\mathrm{M}}, aux)$
$\lambda \leftarrow \mathsf{Inv}_n(1^{\lambda'})$
$\tilde{aux} \leftarrow \overline{\mathrm{M}}$
$(\tilde{\mathrm{M}}_0, \tilde{\mathrm{M}}_1, vk, z) \leftarrow aux$
$x \leftarrow\!\$\ \mathcal{D}(1^{\lambda}, \tilde{\mathrm{M}}_0, \tilde{\mathrm{M}}_1, \tilde{aux})$
$(\mathrm{M}, 1^t, m, \sigma) \leftarrow x$
$d_0 \leftarrow (\tilde{\mathrm{M}}_0(x) \neq \tilde{\mathrm{M}}_1(x))$
$d_1 \leftarrow (\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^{\lambda},vk})$
If $(d_0 \wedge d_1 \wedge m \geq z)$
Then return 1
Else return 0
```

We now show that $\mathsf{S}_0^{\mathsf{tm}}, \mathsf{S}_1^{\mathsf{tm}} \in \boldsymbol{S}^{\mathsf{tm}}_{\mathsf{eq}}$, meaning that these samplers produce functionally equivalent TMs. Both samplers return TMs $\mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}$ and $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^{\lambda},sk^*,vk,m',b}$ that are padded to size $s_1(\lambda)$. First, observe that $\mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}$ contains a signing key sk for DS, whereas $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^{\lambda},sk^*,vk,m',b}$ contains the corresponding punctured signing key sk^*, punctured at $m^* = \langle m' \rangle_{\ell(n(\lambda))}$, and a bit b that equals $\mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}(m')$. According to the definition of a consistent puncturable digital signature scheme, keys sk and sk^* produce the same signatures for all $m \in \mathsf{DS.In}(n(\lambda)) \setminus \{m^*\}$. Note that both $\mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}$ and $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^{\lambda},sk^*,vk,m',b}$ compute a signature for an $\ell(n(\lambda))$-bit string $\langle \overline{\mathrm{M}} \rangle_{\ell(n(\lambda))}$ that is built from the $\ell(\lambda)$-bit input string $\overline{\mathrm{M}}$ by padding it with leading zeros, which is an injective padding. Since m^* can only be built by padding m', these TMs are equivalent for all inputs in $\overline{\mathrm{M}} \in \{0,1\}^{\ell(\lambda)} \setminus \{m'\}$. Furthermore, notice that $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^{\lambda},sk^*,vk,m',b}$ returns $b = \mathrm{M}^{\mathsf{aux}}_{1^{\lambda},sk,vk}(m')$ on input m', so these TMs are equivalent for *all* inputs.

For any $b \in \{0,1\}$ consider game $\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_b^{\mathsf{tm}}}(n(\lambda))$. Let i_b denote the value of i sampled by TM sampler $\mathsf{S}_b^{\mathsf{tm}}$. For any $i \in \{0,1,\ldots,2^{\ell(\lambda)}-1\}$ we have $\Pr[i_b = i] = 2^{-\ell(\lambda)}$, and hence

$$\begin{aligned}\mathsf{Adv}^{\mathsf{io}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_b^{\mathsf{tm}},\mathcal{O}}(n(\lambda)) &= \sum_{i=0}^{2^{\ell(\lambda)}-1} \Big(\Pr[i_b = i] \cdot \Pr[\,\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_b^{\mathsf{tm}}}(n(\lambda)) \,|\, i_b = i\,]\Big) \\ &= 2^{-\ell(\lambda)} \cdot \sum_{i=0}^{2^{\ell(\lambda)}-1} \Pr[\,\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_b^{\mathsf{tm}}}(n(\lambda)) \,|\, i_b = i\,]. \end{aligned} \tag{26}$$

Finally, observe that for any $i \in \{0,1,\ldots,2^{\ell(\lambda)}-1\}$ we have the following by construction:

$$\begin{aligned}\Pr[\,\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_0^{\mathsf{tm}}}(n(\lambda)) \,|\, i_0 = i\,] &= \Pr[\mathrm{G}_{1,i}(\lambda)] - \Pr[\mathrm{G}_{1,i,A}(\lambda)], \\ \Pr[\,\mathrm{IO}^{\mathcal{O}}_{\mathsf{Obf}^{\mathsf{tm}}_{\mathrm{eq}},\mathsf{S}_1^{\mathsf{tm}}}(n(\lambda)) \,|\, i_1 = i\,] &= \Pr[\mathrm{G}_{1,i,B}(\lambda)] - \Pr[\mathrm{G}_{1,i+1}(\lambda)].\end{aligned}$$

Claim 2 follows from Equation (26) together with the two equations above.

<u>Proof of Claim 3.</u> We build a PT adversary $\mathcal{U}$ against the PSUFCMA-security of DS such that for all $\lambda \in \mathbb{N}$ we have $\sum_{i=0}^{2^{\ell(\lambda)}-1} \left(\Pr\left[\,\mathrm{G}_{1,i,A}(\lambda)\,\right] - \Pr\left[\,\mathrm{G}_{1,i,B}(\lambda)\,\right]\right) \leq 2^{\ell(\lambda)+1} \cdot \mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(n(\lambda))$.

```
Adversary $\mathcal{U}_1(1^{\lambda'})$
    $\lambda \leftarrow \mathsf{Inv}_n(1^{\lambda'})$
    $m' \leftarrow\$ \{0,1\}^{\ell(\lambda)}$
    $m^* \leftarrow \langle m' \rangle_{\ell(n(\lambda))}$
    $st \leftarrow m'$
    Return $(m^*, st)$

Adversary $\mathcal{U}_2(1^{\lambda'}, st, vk, sk^*)$
    $\lambda \leftarrow \mathsf{Inv}_n(1^{\lambda'})$ ; $m' \leftarrow st$ ; $b \leftarrow\$ \{0,1\}$
    $hk \leftarrow\$ \mathsf{H.Kg}(1^\lambda)$ ; $h \leftarrow \mathsf{H.Ev}(1^\lambda, hk, \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
    $\mathrm{M}_0 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^0)$ ; $\mathrm{M}_1 \leftarrow \mathsf{Pad}_{s_0(\lambda)}(\mathrm{M}^1_{1^\lambda,hk,h})$
    $aux \leftarrow\$ \mathsf{Obf}^{\mathsf{tm}}_{\mathsf{eq}}(1^{n(\lambda)}, \mathsf{Pad}_{s_1(\lambda)}(\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}))$
    $x \leftarrow\$ \mathcal{D}(1^\lambda, \mathrm{M}_0, \mathrm{M}_1, aux)$ ; $(\mathrm{M}, 1^t, m, \sigma) \leftarrow x$
    $d_0 \leftarrow (\mathrm{M}_0(x) \neq \mathrm{M}_1(x))$ ; $d_1 \leftarrow (\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk})$
    If $(d_0 \wedge d_1 \wedge m = m')$ then return $\sigma$ else return $\perp$
```

Consider the value m' sampled by $\mathcal{U}_1$ in game $\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(n(\lambda))$. For any $i \in \{0, 1, \ldots, 2^{\ell(\lambda)} - 1\}$ it holds that $\Pr[m' = i] = 2^{-\ell(\lambda)}$. Hence,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{psufcma}}_{\mathsf{DS},\mathcal{U}}(n(\lambda)) &= \sum_{i=0}^{2^{\ell(\lambda)}-1} \big(\Pr[m' = i] \cdot \Pr[\,\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(n(\lambda)) \,|\, m' = i\,]\big) \\ &= 2^{-\ell(\lambda)} \cdot \sum_{i=0}^{2^{\ell(\lambda)}-1} \Pr[\,\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(n(\lambda)) \,|\, m' = i\,]. \qquad (27)\end{aligned}$$

Now observe that for any $i \in \{0, 1, \ldots, 2^{\ell(\lambda)} - 1\}$ we also have

$$\Pr[\,\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(n(\lambda)) \,|\, m' = i\,] \geq \frac{1}{2} \cdot \left(\Pr\left[\,\mathrm{G}_{1,i,A}(\lambda)\,\right] - \Pr\left[\,\mathrm{G}_{1,i,B}(\lambda)\,\right]\right). \qquad (28)$$

Let $x = (\mathrm{M}, 1^t, m, \sigma)$ be an output of adversary $\mathcal{D}$ in games $\mathrm{G}_{1,i,A}(\lambda)$ and $\mathrm{G}_{1,i,B}(\lambda)$ (note that the input distribution of $\mathcal{D}$ is the same in both games). If these games produce different outcomes for the same x, it means that $\mathrm{M}^0(x) \neq \mathrm{M}^1_{1^\lambda,hk,h}(x)$, $\mathrm{M} = \mathrm{M}^{\mathsf{ver}}_{1^\lambda,vk}$ and $m = i$. According to the construction of M^0 and $\mathrm{M}^1_{1^\lambda,hk,h}$ it follows that $(\langle m \rangle_{\ell(n(\lambda))}, \sigma)$ is a valid message-signature pair for the digital signature scheme DS with verification key vk.

Whenever the above happens, adversary $\mathcal{U}$ wins in game $\mathrm{PSUFCMA}^{\mathcal{U}}_{\mathsf{DS}}(n(\lambda))$ by forging a valid signature σ for message m^*, as long as the following two conditions are satisfied. First, it is only true if adversary $\mathcal{U}$ sampled $m' = i$. Second, in order to build TM $\mathrm{M}^{\mathsf{aux\text{-}punc}}_{1^\lambda,sk^*,vk,m',b}$, adversary $\mathcal{U}$ has to compute $b = \mathrm{M}^{\mathsf{aux}}_{1^\lambda,sk,vk}(m')$. Since $\mathcal{U}$ does now know sk, instead it has to correctly guess the value of $b \in \{0, 1\}$. Therefore, $\mathcal{U}$ can perfectly simulate the games with probability $\frac{1}{2}$.

Claim 3 follows from Equation (27) and Equation (28). ▌

Acknowledgments

We thank the Eurocrypt 2016 reviewers for their comments.

References

[1] P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`. 2, 5, 9

[2] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. From selective to adaptive security in functional encryption. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, Aug. 2015. 6

[3] B. Applebaum, B. Barak, and A. Wigderson. Public-key cryptography from different assumptions. In L. J. Schulman, editor, *42nd ACM STOC*, pages 171–180. ACM Press, June 2010. 5

[4] M. Backes, O. Dagdelen, M. Fischlin, S. Gajek, S. Meiser, and D. Schröder. Operational signature schemes. Cryptology ePrint Archive, Report 2014/820, 2014. `http://eprint.iacr.org/2014/820`. 6

[5] M. Backes, S. Meiser, and D. Schröder. Delegatable functional signatures. Cryptology ePrint Archive, Report 2013/408, 2013. `http://eprint.iacr.org/2013/408`. 4, 6

[6] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014. 2

[7] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001. 2, 3, 9

[8] M. Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–284, 2002. 5, 7

[9] M. Bellare and G. Fuchsbauer. Policy-based signatures. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 520–537. Springer, Heidelberg, Mar. 2014. 4, 6

[10] M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 470–484. Springer, Heidelberg, Aug. 1997. 8

[11] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 7

[12] M. Bellare, I. Stepanovs, and S. Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 102–121. Springer, Heidelberg, Dec. 2014. 2, 6, 9

[13] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the existence of extractable one-way functions. In D. B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. 6

[14] N. Bitansky, S. Garg, H. Lin, R. Pass, and S. Telang. Succinct randomized encodings and their applications. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015. 6

[15] N. Bitansky and O. Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 241–250. ACM Press, June 2013. 2, 3

[16] N. Bitansky, O. Paneth, and D. Wichs. Perfect structure on the edge of chaos. Cryptology ePrint Archive, Report 2015/126, 2015. `http://eprint.iacr.org/2015/126`. 6

[17] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, Dec. 2013. 4, 8

[18] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, Feb. 2014. 2, 5, 6, 9

[19] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, Mar. 2014. 4, 6, 8

[20] E. Boyle and R. Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/2013/703`. 6

[21] C. Brzuska and A. Mittelbach. Using indistinguishability obfuscation via UCEs. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 122–141. Springer, Heidelberg, Dec. 2014. 6

[22] R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015. 6

[23] R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, Mar. 2015. 5

[24] N. Chandran, S. Raghuraman, and D. Vinayagamurthy. Constrained pseudorandom functions: Verifiable and delegatable. Cryptology ePrint Archive, Report 2014/522, 2014. `http://eprint.iacr.org/2014/522`. 6

[25] G. Fuchsbauer. Constrained verifiable random functions. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 95–114. Springer, Heidelberg, Sept. 2014. 6

[26] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, Oct. 2013. 2, 9

[27] S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, Aug. 2014. 2, 3, 14, 15, 17

[28] C. Gentry, S. Halevi, M. Raykova, and D. Wichs. Outsourcing private RAM computation. In *55th FOCS*, pages 404–413. IEEE Computer Society Press, Oct. 2014. 6

[29] C. Gentry, A. Lewko, A. Sahai, and B. Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/2014/309`. 2

[30] C. Gentry, A. B. Lewko, and B. Waters. Witness encryption from instance independent assumptions. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 426–443. Springer, Heidelberg, Aug. 2014. 5

[31] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, Oct. 1986. 8

[32] S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *46th FOCS*, pages 553–562. IEEE Computer Society Press, Oct. 2005. 2, 3

[33] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988. 3

[34] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. 5

[35] S. Hada. Zero-knowledge and code obfuscation. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 443–457. Springer, Heidelberg, Dec. 2000. 2, 3

[36] I. Haitner, T. Holenstein, O. Reingold, S. P. Vadhan, and H. Wee. Universal one-way hash functions via inaccessible entropy. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 616–637. Springer, Heidelberg, May 2010. 8

[37] Y. Ishai, O. Pandey, and A. Sahai. Public-coin differing-inputs obfuscation and its applications. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, Mar. 2015. 6

[38] J. Katz and C.-Y. Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005. `http://eprint.iacr.org/2005/328`. 8

[39] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, Nov. 2013. 4, 8

[40] V. Koppula, A. B. Lewko, and B. Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015. 6

[41] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 8

[42] R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, Heidelberg, Aug. 2014. 2

[43] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. 8, 18

[44] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. 2, 6, 9, 10

[45] V. Shoup. A composition theorem for universal one-way hash functions. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 445–452. Springer, Heidelberg, May 2000. 8, 18

[46] B. Waters. A punctured programming approach to adaptively secure functional encryption. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, Aug. 2015. 6