Impossibility on Tamper-Resilient Cryptography with Uniqueness Properties

Yuyu Wang[1] ⋆, Takahiro Matsuda[2], Goichiro Hanaoka[2], and Keisuke Tanaka[3]

[1] University of Electronic Science and Technology of China, Chengdu, China
wangyuyu@uestc.edu.cn
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
t-matsuda@aist.go.jp, hanaoka-goichiro@aist.go.jp
[3] Tokyo Institute of Technology, Tokyo, Japan
keisuke@is.titech.ac.jp

Abstract. In this work, we show negative results on the tamper-resilience of a wide class of cryptographic primitives with uniqueness properties, such as unique signatures, verifiable random functions, signatures with unique keys, injective one-way functions, and encryption schemes with a property we call unique-message property. Concretely, we prove that for these primitives, it is impossible to derive their (even extremely weak) tamper-resilience from any common assumption, via black-box reductions. Our proofs exploit the simulatable attack paradigm proposed by Wichs (ITCS '13), and the tampering model we treat is the plain model, where there is no trusted setup.

Keywords. black-box separation, simulatable attack, tamper-resilience, uniqueness

1 Introduction

1.1 Background

Motivated by the fact that an adversary may maliciously modify the secret information of a cryptographic scheme by executing tampering attacks (e.g., heating up devices or injecting faults [12,13]) and observe the effect of the changes, Bellare and Kohno [10] and Gennaro et al. [37] independently initiated the study on tamper-resilient primitives. Bellare and Kohno proposed block-cipher against restricted tampering attacks (i.e., the class of tampering functions used by the adversary is restricted[4]), and gave a negative result showing that there exists no tamper-resilient block-cipher against arbitrary tampering functions. In their model (called the plain model in our paper), secret keys are potentially tampered with and there is no trusted setup. Gennaro et al. treated primitives against

⋆ ORCID: 0000-0002-1198-1903. Research was conducted at Tokyo Institute of Technology.

[4] Here, tampering functions mean functions used by adversaries that take as input original keys and output tampered keys.

arbitrary tampering functions, whereas secret keys implicitly contain trusted public keys in their model (called the on-line model in [35]). Although most following works [7,8,49,11,53,67,23,24,62,34,28,35,17] did not adopt the on-line model, they assumed the existence of trusted common reference strings (CRSs), due to the difficulty of achieving tamper-resilience in the plain model. Amongst them, several works [24,23,28,35,17] are secure even when tampering functions could be arbitrary. Such strong security notions are worth considering since it is hard to restrict the range of attacks in practice. However, it is not desirable to put a strong trust on the entity that sets up public parameters in practice, and the assumption that tamper-proof public parameters are available is very strong. Especially, an adversary, who can execute fault attacks on secret keys, should also be able to alter CRSs stored together with the keys in the device. Therefore, it is desirable to understand what kind of primitives can be tamper-resilient in the plain model. The research by Dziembowski et al. [27] showed us a promising way to achieve tamper-resilience in the plain model, which utilizes non-malleable codes. Indeed, combining non-malleable codes with standard primitives can straightforwardly derive primitives (even with uniqueness properties) secure against (at least one-time) tampering attacks in the plain model. However, this work requires restrictions on tampering functions (and so do the following works (e.g., [29,30,46,18,6,22])). Another work by Ateniese et al. [4] proved that unique signatures are secure against subversion attacks, which allow an adversary to maliciously modify signing algorithms and hence capture tampering attacks. However, their results assume that the attacks meet an undetectability property or cryptographic reverse firewalls are available.

Up until now, all the (positive or negative) results on tamper-resilient primitives against arbitrary tampering functions either assumed tamper-proof public parameters (e.g., [49,23,24,28,35]), or focused on symmetric cryptography [10]. Hence, it remains unclear whether public key primitives can achieve (full) tamper-resilience in the plain model. In this paper, we study public key cryptography in this model and show broad negative results.

1.2 Our Results

We focus on the impossibility of proving the tamper-resilience of a wide class of cryptographic primitives via black-box reductions, and show several negative results. The type of black-box reduction we consider is the so-called fully black-box reduction, which does not use the code of adversaries. Moreover, a reduction algorithm should break the underlying assumption as long as the utilized adversary successfully breaks the tamper-resilience, no matter how much computing power the adversary has. We remark that most cryptographic primitives are proved to be secure via such type of reductions. We detail our results below.

Impossibility on provable deterministic primitives (PDPs) and signatures. At first, we consider a negative result on a class of primitives called PDPs, which were firstly defined by Abe et al. [2]. A PDP evaluates a function by using a secret key and generates a proof for the input/output pair, and

it is required that there exist only one valid output for each input. The definition of PDPs captures various primitives such as verifiable random functions (VRFs) [55,54,44], verifiable unpredictable functions (VUFs) [55,54], and unique signatures [39,55,54]. The (perfect) uniqueness property of these primitives prevents malicious signers from easily outputting many signatures on the same input, and thus prevents a simple denial-of service attack on a verifier forced to verify many outputs on the same input, even when the key pairs are selected by the signer in a subtle way. Also, due to this property, PDPs can be viewed as perfectly binding commitments to an exponential number of (perhaps random-looking) outputs [54], and thus can play important roles in micropayments [57], resettable zero-knowledge proofs [56], updatable zero-knowledge databases [52], verifiable transaction escrow schemes [47], etc. On account of the wide usage of PDPs, their security under tampering attacks is important and worth studying.

For these primitives, we show that it is impossible to achieve their tamper-resilience via black-box reductions. More specifically, we show that if a PDP is weakly unpredictable,[5] then there exists no black-box reduction deriving its tamper-resilience from any assumption captured by the notion of a cryptographic game [40,38,68], where a (possibly inefficient) challenger interacts with a monolithic adversary.[6] Here, weak unpredictability only requires that any probabilistic polynomial-time (PPT) adversary, neither allowed to make a query nor given a public key, cannot come up with a valid input/output pair. It is clear that any non-trivial PDP should satisfy such weak security. Furthermore, differently from negative results in [37,35], we treat very weak tamper-resilience, where an adversary only makes one tampering query and one computing query.[7] Hence, our result also captures tamper-resilient PDPs with self-destructive or key-updating mechanism.[8] We prove our result by using the simulatable attack paradigm proposed by Wichs [68], which is discussed in more details in the next subsection. By slightly modifying this proof, we can extend our negative result for a more general notion called re-randomizable signatures [66,43].

As by-product results, we prove the same negative result on weakly unforgeable unique-key signatures, in which there exists only one valid secret key for each public key, and injective one-way functions (OWFs). Here, weak unforgeability is defined in the same way as weak unpredictability, and up until now, a broad class of existing signature schemes, e.g., ones where secret keys are discrete logarithms or factoring of public keys, or key pairs are in the Diffie-Hellman form,

[5] Unless explicitly stated otherwise, when referring to weak unpredictability, we mean computational weak unpredictability, not statistical weak unpredictability. The same argument is made for other security notions.

[6] By a monolithic adversary, we mean an adversary that is a single entity. Its antonym is a "multi-stage" adversary that consists of two or more components among which the state information cannot be passed freely [63].

[7] When focusing on negative results, the defined tamper-resilience is desirable to be as weak as possible.

[8] Self-destructive mechanism prevents an adversary from learning information by making further queries when tampering is detected, and key-updating mechanism allows a device to update its secret information.

is captured by the notion of unique-key signatures (e.g., [36,21,51,66,45]). These results not only show the reason why many existing schemes cannot be proven tamper-resilient, but also indicate that when constructing signatures and OWFs in the presence of tampering attacks, one should circumvent the unique-key property and injectiveness. Note that there is no contradiction between our work and tamper-resilient unique signatures (implicitly) implied by previous results on non-malleable codes or subversion-resilient signatures [27,29,30,46,18,6,4], since those results require either restrictions on tampering functions or reverse firewalls as mentioned before.

Impossibility on encryption schemes. Next, we give a negative result on a class of public key encryption (PKE) schemes that we call *unique-message* PKE schemes, where for a ciphertext (possibly outside the support of the encryption algorithm), all the valid secret keys with respect to a public key (possibly outside the support of the key generation algorithm) lead to the same decryption result. More specifically, for a unique-message PKE scheme, in addition to ordinary algorithms as a PKE scheme, we require that there be an algorithm that we call the "plaintext-recovering" algorithm. Its syntax is exactly the same as the ordinary decryption algorithm. We require that it satisfy the usual correctness as the decryption algorithm. What makes it different from the ordinary decryption algorithm is that for each public key and each ciphertext, it holds that the decryption results are the same for all valid secret keys with respect to the public key (see Definition 15 for the formal definition). We note that the plaintext-recovering algorithm may be the original decryption algorithm, but in general it need not be so.

Our negative result shows that if a unique-message PKE scheme is weakly one-way, then there exists no black-box reduction deriving its tamper-resilience from any assumption captured by the notion of a *restricted* cryptographic game (i.e., any common falsifiable assumption [59,38]), where the challenger is restricted to be PPT. Here, weak one-wayness is defined in the same way as standard one-wayness, except that we only treat adversaries that are not allowed to see the public key. In other words, it only requires that any PPT adversary, neither allowed to make decryption queries nor given a public key, cannot recover the message from a randomly generated ciphertext. This is clearly a very weak security notion, and should be satisfied by any non-trivial PKE scheme. Furthermore, similarly to the cases of PDPs and signatures, we consider very weak tamper-resilience, where an adversary only makes one tampering query and one decryption query. Unlike our result for PDPs, this result does not capture black-box reductions to non-falsifiable assumptions where challengers are computationally unbounded, unless we assume that the PKE scheme is statistically weakly one-way. However, statistical weak one-wayness is not necessarily implied by (computational) one-wayness. We will give more details in Section 4.2. Here, notice that [23] also shows a negative result with respect to adversaries making a single tampering and decryption query. However, it only treats a strong type of tampering queries called "post-challenge" tampering queries, which can be dependent on the challenge ciphertext. Their negative result can be circum-

vented when considering challenge-independent tampering queries which capture a-prior tampering attacks in practice. Moreover, their negative result holds only for indistinguishability against chosen ciphertext attacks. It was unclear whether tamper resilient one-wayness against (even weak) chosen ciphertext attacks is achievable, while we give a partial but strong negative answer.

Although the definition of a unique-message PKE scheme has never been formalized before, it captures many naturally constructed PKE schemes (e.g., [64,36,20,15,41]). More specifically, since a PKE scheme is required to satisfy correctness, all the valid ciphertexts (i.e., ones in the support of the encryption algorithm) should be decrypted to a unique message, while invalid ciphertexts usually lead the decryption algorithm to abort, in order to prevent adversaries from learning useful information from the answers of decryption queries. Hence, a ciphertext is typically decrypted to a unique message if the used secret key is correct. However, due to possible difficulties of directly proving some implementations to be a unique-message PKE scheme, instead of insisting that the original decryption algorithm decrypt any ciphertext to a unique message, we give a relaxation by introducing the notion of the plaintext-recovering algorithm mentioned in the above paragraph. For ease of understanding the rationale behind our definition, as an instance, we show how it captures the Cramer-Shoup scheme [20] in Section E.1 in Supplementary Material. Furthermore, one can also see that unique-key PKE schemes, where there exists only one valid secret key for each public key, can be cast as unique-message PKE schemes (see Section E.2 in Supplementary Material). Similar to our negative results for signatures, our results for PKE schemes clearly show the exact barrier when proving tamper-resilience for existing schemes, and also give a guideline for future works on tamper-resilient PKE schemes.

Remark on the plain model. Our research focuses on tamper-resilient primitives in the plain model. This model might seem strong and assuming the existence of tamper-proof public parameters may be reasonable to some extent. However, such an assumption is not always realistic since fault attacks allow adversaries to maliciously modify CRSs stored in the devices. Besides, another line of works on subversion-resilient non-interactive proof systems (e.g., [9,1,33]) also alerted the danger of trusting CRSs. Therefore, in addition to previous positive results mentioned before [27,29,30,46,18,6,4], it is desirable to deepen the understanding of tamper-resilient primitives in the plain model. Moreover, we believe that trying to circumvent our negative results is a good starting point for future positive works, as we will mention in the open problems in Section 5.

1.3 High-Level Idea and Technique

Tamper-resilient primitives are secure against adversaries that try to tamper with the secret key, make (computing, signing, or decryption) queries, and output a forgery or recover a message. To prove the tamper-resilience of a primitive under some cryptographic assumption via a (fully) black-box reduction, one

needs to construct a reduction algorithm that has access to any successful adversary against the tamper-resilience and breaks the assumption. Therefore, to show the impossibility on black-box reductions from tamper-resilience to common assumptions, what we need to do is to rule out the existence of reduction algorithms which can obtain useful information from any successful adversary, in any cryptographic game. Our basic idea for proving this is to show that any reduction algorithm cannot answer the queries made by some successful adversary, or it cannot benefit from the outputs of the adversary.

Tamper-resilience model. Before describing how we achieve our goal, we describe how we model a valid adversary against tamper-resilience in more details. Such an adversary consists of three independent components $(\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$, which are allowed to share states before the security game but do not have communicating tapes once the game begins.[9] **Tamper** models a tampering function that on input the original secret key tries to output some tampered key helpful to $(\mathbf{Break}1, \mathbf{Break}2)$. **Break**1 makes a (computing, signing, or decryption) query. On input the answer generated via the tampered key, **Break**2 tries to output some forgery (or recover a message from the challenge ciphertext). The model we consider captures the very weak type of tampering attacks where the adversary can only make one selectively determined tampering query, which in turn makes our negative results very strong. Notice that $(\mathbf{Break}1, \mathbf{Break}2)$ learns no information on the secret key other than the information leaked from the answer of the, say, computing query, since $(\mathbf{Break}1, \mathbf{Break}2)$ and **Tamper** (which sees the secret key) do not share any state generated during the security game. It is the same as the case that an adversary $(\mathbf{Break}1, \mathbf{Break}2)$ determines the way to tamper with the secret key at the beginning of the game and tries to benefit from the answer of the computing query. We justify our model as follows.

Remark on our model. In our model, we define the tampering function as part of the adversary and the reduction can only have black-box access to it, while in previous works [24,23,28,35] considering arbitrary tamper attacks, the tampering functions are defined as tampering queries made by the adversary. Therefore, one may wonder whether the reductions considered by us are more restricted. We stress that in the security proofs of all these works, the reductions also access the functions in a black-box manner. Indeed, those reductions sometimes change the functions by hard-wiring other functions or values in the tampering functions, which make the functions seem "non-black-box". However, such procedures can be treated as changing the input of the functions rather than exploiting the structure of the functions themselves. Therefore, such modification for tampering functions is also allowed in our model. For instance, the reduction can query the hard-wired function to its challenger and give the answer back to **Tamper** as its input. As a result, our model does not make any additional restrictions on the reduction.

[9] We forbid the communication between **Break**1 and **Break**2 for simplicity, and such restriction makes our results stronger since we focus on negative results.

We also stress that we can model an adversary in the way that $(\mathbf{Break}1, \mathbf{Break}2)$ sends a (selectively determined) tampering function **Tamper** to modify a secret key sk to a tampered key $\mathbf{Tamper}(sk)$ as in previous works (e.g., [24,23,28,35]), while for a fully black-box reduction, which only has black-box access to **Tamper**, this does not make any difference. We define **Tamper** as part of the adversary only for simplicity. Our model does not rule out reductions exploiting the structures of arbitrary tampering functions (rather than restricted ones), while it would be surprising if there would be any.

Our intuition. We now describe an intuition behind our proofs. For ease of understanding, we talk about the case of unique signatures. Let $\mathcal{R}$ be a reduction algorithm trying to attack some underlying assumption captured by a typical cryptographic game with access to a successful adversary $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$. To benefit from $\mathcal{A}$, $\mathcal{R}$ has to answer the signing query made by **Break**1. However, we observe that if $\mathcal{R}$ has not given **Tamper** a valid secret key previously,[10] it may have no idea what the tampered key is, in which case *answering the signing query is as difficult as breaking the underlying security of the unique signature scheme for* $\mathcal{R}$. On the other hand, if $\mathcal{R}$ has given a valid key to **Tamper**, then it is able to forge a signature by itself. In this case, since *a signature forged by* **Break**2 *must be the same as the one forged by* $\mathcal{R}$ *due to uniqueness, it does little help to* $\mathcal{R}$. As a result, the access to $\mathcal{A}$ may not benefit $\mathcal{R}$ in the cryptographic game, which gives us the conflict. Note that when formally showing the existence of such a successful but "useless" adversary $\mathcal{A}$, we need to take care of more details (e.g., the way for $\mathcal{A}$ to check the validity of a secret key without having a key checking algorithm), which we do not mention here for simplicity.

The above intuition is also adopted to show negative results on VRFs, VUFs, unique-key signatures, injective OWFs, and unique-message PKE schemes, while in the case of unique-message PKE schemes, our proof exploits the difficulty of answering decryption queries (instead of signing queries). Like the case of unique signatures, the uniqueness properties of these primitives ensure that the outputs of $\mathcal{A}$ and $\mathcal{R}$ are identical.

We formalize our intuition by using meta-reductions, which have appeared in a great deal of previous works (e.g., [14,19,60,32,38,61,3,68,69,5,50,58,42,22,31]). Roughly speaking, we firstly give an inefficient (but valid) adversary $\mathcal{A}$ breaking the tamper-resilience of a class of schemes, and then a PPT algorithm **Sim** (which does not necessarily have the structure of a valid adversary) simulating the action of $\mathcal{A}$. For any black-box reduction $\mathcal{R}$ deriving the tamper-resilience of these schemes from some cryptographic game $\mathcal{G}$, we could obtain a PPT adversary $\mathcal{R}^{\mathbf{Sim}}$ breaking $\mathcal{G}$ with the same advantage as $\mathcal{R}^{\mathcal{A}}$, which is infeasible if $\mathcal{G}$ is hard to break. Hence, we can derive the non-existence of $\mathcal{R}$. More specifically, we give our proof under the simulatable attack paradigm, a meta-reduction method proposed by Wichs [68]. Most negative results proved using meta-reductions

[10] Here, a valid secret key means a secret key passing the key checking procedure executed by **Tamper**.

implicitly fall under this paradigm, and recent works [16,26] explicitly used this paradigm to show negative results on entropic search learning with errors and extractors for entropy extractor-dependent sources. What we have to do is to show that any (possibly inefficient) oracle-access machine cannot distinguish $\mathcal{A}$ and **Sim**. By doing this we can separate the tamper-resilience of our target schemes from any cryptographic game $\mathcal{G}$ rather than some particular one. We remark that when proving the impossibility result on unique-message PKE schemes, the oracle-access machine is required to be efficient, in which case we separate the tamper-resilience from any *restricted* cryptographic game. We refer the reader to Sections 3.3 and 4.3 for the details of our adversaries and simulators and the ideas behind their constructions.

Comparison with previous works. Finally, let us highlight the differences between the work by Wichs and ours. In [68], Wichs showed that there is no black box reduction for proving the security of leakage-resilient unique witness one-way relations, leaky pseudo-entropy generators, entropy condensers, and correlation-resilient OWFs from the assumptions captured by cryptographic games.[11] His negative results for the latter two primitives are vastly different stories, since the adversaries are modeled in different ways, while those for the former two are more related to our results, in the sense that both leakage-resilient and tamper-resilient primitives prevent attacks on memory. However, it should be noticed that leakage-resilience treats adversaries directly obtaining leaked information, while tamper-resilience treats ones observing the effect of malicious modifications. Since the restrictions on adversaries are completely different, our results and those by Wichs do not imply each other, and designing inefficient adversaries and PPT simulators in our case is never easy. Furthermore, our proofs utilize a new methodology that proves the indistinguishability between adversaries and simulators based on computational security assumptions. This is quite different from Wichs' results since the proofs in his work do not use them.

Other related negative results on unique primitives. Coron [19] showed impossibility on simple reductions deriving the tight security of unique sigantures from non-interactive assumptions. Later Kakvi and Kiltz [48] fixed Coron's result by giving a stricter definition of unique signatures. Hofheinz et al. [43] extended the negative results in [19,48] for the notion of re-randomizable signatures, which is more general and hence captures more instantiations, such as the Waters signature scheme [66] and its variant [43]. Recently, Morgan and Pass [58] proposed an impossibility result by ruling out any linear-preserving black-box reduction deriving the security of unique signatures from bounded-round assumptions. In another line, Wang et al. [65] ruled out memory-tight black-box reductions deriving the multi-challenge security of signatures from any computational assumption.

[11] Interestingly, in [68], Wichs also mentioned a negative result on leakage-resilient unique signatures.

1.4 Outline of This Paper

In Section 2, we recall the definitions of cryptographic games (and properties) and simulatable attacks. In Section 3, we give our negative results on PDPs, unique-key signatures, and injective OWFs. In Section 4, we give our negative results on unique-message PKE schemes. In Section 5, we discuss open problems.

2 Preliminaries

In this section, we review several definitions and terminologies that are necessary to describe our results.

Notation. *negl* denotes an unspecified negligible function. If $\mathcal{X}$ is a finite set, then $x \leftarrow \mathcal{X}$ denotes the process of uniformly sampling x at random from the set $\mathcal{X}$. If $\mathcal{A}$ is a deterministic (respectively, probabilistic) algorithm, then $y = \mathcal{A}(x)$ (respectively, $y \leftarrow \mathcal{A}(x)$) means that $\mathcal{A}$ on input x outputs y. Letting the internal randomness space of a probabilistic algorithm $\mathcal{A}$ be $\mathcal{R}_a$, computing $y \leftarrow \mathcal{A}(x)$ is equivalent to sampling $r \leftarrow \mathcal{R}_a$ and then computing $y = \mathcal{A}(x; r)$.

2.1 Cryptographic Game (Property)

In this subsection, we recall the definitions of a cryptographic game and a cryptographic property.

Definition 1 (Cryptographic game [40]). *A* cryptographic game $\mathcal{G}$ *consists of a (possibly inefficient) random system (called the challenger)* $\mathcal{CH}$ *and a constant* $c \in [0, 1)$*. For some security parameter* 1^λ*,* $\mathcal{CH}(1^\lambda)$ *interacts with some adversary* $\mathcal{A}(1^\lambda)$*, and outputs a bit* b*. This interaction is denoted by* $b \leftarrow (\mathcal{A}(1^\lambda) \leftrightharpoons \mathcal{CH}(1^\lambda))$*, and the advantage of* $\mathcal{A}$ *in* $\mathcal{G}$ *is* $\mathbf{Adv}^{\mathcal{A}}_{\mathcal{G}}(\lambda) = \Pr[1 \leftarrow (\mathcal{A}(1^\lambda) \leftrightharpoons \mathcal{CH}(1^\lambda))] - c$*.*

A cryptographic game $\mathcal{G}$ *is secure if for any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathcal{A}}_{\mathcal{G}}(\lambda) \leq negl(\lambda)$*.*

As noted in [38,68], all commonly used assumptions in cryptography fall under the framework of cryptographic games.

A restricted version of the above definition, which only considers PPT challengers and captures most *falsifiable* assumptions [59,38], is given as follows.

Definition 2 (Restricted cryptographic game [38]). *A* restricted cryptographic game *is defined in exactly the same way as a cryptographic game, except that we replace "(possibly inefficient) random system" with* "PPT random system"*.*

Cryptographic property. As noted by Wichs [68], although the definition of a cryptographic game captures all common assumptions, it does not capture some cryptographic properties against stateless adversaries consisting of multiple independent components (e.g., leakage-resilience of one-way relations defined in [68] and tamper-resilience of PDPs, signatures, and PKE schemes defined later in our paper). Following [68], we give a very general definition of an arbitrary cryptographic property $\mathcal{P}$ and use $\mathbf{Adv}_{\mathcal{P}}^{\mathcal{A}_\lambda}(\lambda)$ to denote the advantage of $\mathcal{A}_\lambda$ in breaking a cryptographic property $\mathcal{P}$, where λ is the security parameter. $\mathcal{P}$ is said to be secure if for any PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{P}}^{\mathcal{A}_\lambda}(\lambda)$ is negligible in λ.

2.2 Simulatable Attack

The simulatable attack paradigm is a meta-reduction method formalized by Wichs [68]. Showing the existence of simulatable attacks on cryptographic properties is a general way to show the impossibility of deriving these properties from common assumptions via black-box reductions. We now recall the definitions of a black-box reduction and simulatable attack as follows. Our definition here is based on [68].

Definition 3 (Black-box reduction). *Let $\mathcal{P}$ be some cryptographic property and $\mathcal{G}$ be some cryptographic game. An oracle-access PPT machine $\mathcal{R}^{(\cdot)}$ is said to be a* black-box reduction *deriving the security of $\mathcal{P}$ from $\mathcal{G}$, if for any (possibly inefficient, non-uniform) adversary $\mathcal{A}_\lambda$ such that $\mathbf{Adv}_{\mathcal{P}}^{\mathcal{A}_\lambda}(\lambda) = 1$, there exists a non-negligible function ϵ such that we have $\mathbf{Adv}_{\mathcal{G}}^{\mathcal{R}^{\mathcal{A}_\lambda}}(\lambda) \geq \epsilon(\lambda)$.*

In the above definition, we require $\mathbf{Adv}_{\mathcal{P}}^{\mathcal{A}_\lambda}(\lambda)$ to be 1 (rather than non-negligible), which makes the defined reduction very restrictive. Since our focus is on a black-box separation, the more restricted the type of black-box reductions is, the stronger our negative results become. Furthermore, the definition in [68] is strengthened by requiring that a black-box reduction $\mathcal{R}$ have noticeable advantage in breaking $\mathcal{G}$, while we only require $\mathcal{R}$ to have a non-negligible advantage, which is more common.[12]

A simulatable attack on a cryptographic property $\mathcal{P}$ consists of a valid but possibly inefficient adversary $\mathcal{A}$ and a possibly invalid but efficient simulator $\mathbf{Sim}$.[13] If $\mathcal{A}$ breaks $\mathcal{P}$ and is indistinguishable from $\mathbf{Sim}$ for any oracle-access machine, then the existence of a black-box reduction $\mathcal{R}$ deriving the security of $\mathcal{P}$ from some cryptographic game $\mathcal{G}$ implies that breaking the assumption captured by $\mathcal{G}$ is not hard. This follows from the fact that the success of $\mathcal{R}^{\mathcal{A}}$ implies the success of $\mathcal{R}^{\mathbf{Sim}}$ in $\mathcal{G}$.

Definition 4 (Simulatable attack [68]). *A* simulatable attack *on a cryptographic property $\mathcal{P}$ consists of: (a) an ensemble of (possibly inefficient) stateless*

[12] We note that as discussed in [68], all the proofs given by Wichs can be extended to the case that the advantage of $\mathcal{R}$ is only required to be non-negligible.

[13] In our case, a valid (respectively, invalid) adversary means a stateless (respectively, stateful) adversary.

non-uniform adversaries $\{\mathcal{A}_{\lambda,f}\}_{\lambda\in\mathbb{N},f\in\mathbb{F}_\lambda}$ *where* $\{\mathbb{F}_\lambda\}_\lambda$ *are some finite sets, and (b) a PPT stateful simulator* **Sim**. *Furthermore, the following two properties are required to hold.*

- *For all* $\lambda \in \mathbb{N}$ *and* $f \in \mathbb{F}_\lambda$, $\mathbf{Adv}_{\mathcal{P}}^{\mathcal{A}_{\lambda,f}}(\lambda) = 1$.
- *For all (possibly inefficient) oracle-access probabilistic machines* $\mathcal{B}^{(\cdot)}$ *making at most polynomially many queries to its oracle, we have*

$$|\Pr_{f\leftarrow\mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)]| \leq negl(\lambda).$$

Definition 5 (Weak simulatable attack). *A* weak simulatable attack *is defined in exactly the same way as a simulatable attack, except that we replace "(possibly inefficient) oracle-access probabilistic machines* $\mathcal{B}^{(\cdot)}$*" with "oracle-access PPT machines* $\mathcal{B}^{(\cdot)}$*".*

Note that in the above definitions, $\mathbb{F}_\lambda$ is a set, and an adversary is modeled as an ensemble of algorithms $\mathcal{A}_{\lambda,f}$ where each instance hardwires an element $f \in \mathbb{F}_\lambda$. Note also that the simulator **Sim** is only required to simulate the behavior of adversaries $\mathcal{A}_{\lambda,f}$ in the situation where f is chosen uniformly at random from $\mathbb{F}_\lambda$. Like in [68], we set $\mathbb{F}_\lambda$ as a set of all functions with some specific domain and range when showing our negative results. This ensures that the outputs of f look like real randomness in the view of $\mathcal{B}$, and they can be simulated by $\mathbf{Sim}(1^\lambda)$ by executing lazy sampling. We refer the reader to Sections 3.3 and 4.3 for the details.

The following theorem by Wichs [68] shows that the existence of a (weak) simulatable attack on some cryptographic property $\mathcal{P}$ implies the impossibility of deriving the security of $\mathcal{P}$ from any (restricted) cryptographic game $\mathcal{G}$ via black-box reductions.

Theorem 1 ([68]). *If there exists a simulatable attack (respectively, weak simulatable attack) on some cryptographic property* $\mathcal{P}$ *and a black-box reduction deriving the security of* $\mathcal{P}$ *from the security of some cryptographic game (respectively, restricted cryptographic game)* $\mathcal{G}$*, then there exists some PPT adversary* $\mathcal{A}$ *that has non-negligible advantage in* $\mathcal{G}$.

We refer the reader to [68] for the proof of Theorem 1. Notice that the original proof in [68] did not show that a weak simulatable attack implies the black-box separation with respect to restricted cryptographic games, and it asks the reduction algorithm to have a noticeable advantage. However, extending it to show the above theorem is straightforward.

3 Impossibility on Provable Deterministic Primitives and Unique-Key Signatures

In this section, we give negative results on tamper-resilient PDPs (including tamper-resilient VUFs, VRFs, and unique signatures as special cases) and tamper-resilient unique-key signatures. Our results show that if a PDP or

unique-key signature scheme satisfies some "extremely" weak unpredictability or unforgeability, then there exists no black-box reduction deriving its tamper-resilience in the plain model from any commonly used assumption.

The rest of this section is organized as follows. In Section 3.1, we recall the definitions of PDPs and signatures. In Section 3.2, we define several security notions. In Section 3.3, we show the existence of simulatable attacks on the tamper-resilience of PDPs and signatures. In Section 3.4, we summarize our negative results.

3.1 Definitions of PDPs and Signatures

At first, we recall the definition of a PDP, which is formalized in [2] and captures VUFs, VRFs, and unique signatures as special cases.

Definition 6 (Provable deterministic primitive (PDP) [2]). *A PDP consists of the polynomial-time (PT) algorithms* $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$*. (a)* Gen *is a probabilistic algorithm that takes as input* 1^λ*, and returns a public/secret key pair* $(pk, sk) \in \{0,1\}^p \times \{0,1\}^s$ *for some polynomials* $p = p(\lambda)$ *and* $s = s(\lambda)$*. The set of all secret keys (output by* $\mathsf{Gen}(1^\lambda)$*) and the (internal) randomness space of* Gen *are respectively denoted by* $\mathcal{SK}$ *and* $\mathcal{R}_g$*. (b)* Comp *is a deterministic algorithm that takes as input a secret key* sk *and* $x \in \mathcal{X}$*, where* $\mathcal{X}$ *denotes the domain, and returns some value* y*. (c)* Prove *is a probabilistic algorithm that takes as input a secret key* sk *and* x*, and returns a proof* π*. The (internal) randomness space of* Prove *is denoted by* $\mathcal{R}_p$*. (d)* Verify *is a deterministic algorithm that takes as input a public key* pk*,* x*,* y*, and a proof* π*, and returns* 1 *(accept) or* 0 *(reject).*

A PDP is required to satisfy uniqueness *and* correctness*. Uniqueness is said to be satisfied if for all* $\lambda \in \mathbb{N}$*, all* $pk \in \{0,1\}^p$ *(possibly outside the support of* Gen*) and all* $x \in \mathcal{X}$*, there exists no tuple* (y, π, y', π') *that simultaneously satisfies* $y \neq y'$ *and* $\mathsf{Verify}_{pk}(x, y, \pi) = \mathsf{Verify}_{pk}(x, y', \pi') = 1$*. Correctness is said to be satisfied if* $\mathsf{Verify}_{pk}(x, \mathsf{Comp}_{sk}(x), \mathsf{Prove}_{sk}(x)) = 1$ *holds for all* $\lambda \in \mathbb{N}$*, all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, and all* $x \in \mathcal{X}$*.*

The syntax of a PDP is exactly the same as that of a VRF and a VUF, and it also captures unique signature schemes, which we discuss later in this subsection. Notice that the definition of uniqueness is different from that in [2], which additionally requires the public parameter (including the bilinear group) to be correctly generated since it treats structure-preserving constructions. Other than that we do not consider public parameters separately or some other relaxed uniqueness notions (e.g., uniqueness holding for most public keys), one can see that our definition is equivalent to the original definitions of uniqueness in [55,54]. Such original definitions prevent denial-of service attacks and provide perfect binding properties when PDPs are used as a special type of commitments in their applications (e.g., micropayments [57], resettable zero-knowledge proofs [56], updatable zero-knowledge databases [52], verifiable transaction escrow schemes [47]), even in the presence of maliciously chosen CRSs and public keys.

We now recall the definition of a (digital) signature scheme.

Definition 7 (Digital signature). *A signature scheme consists of the PT algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*. (a)* Gen *is a probabilistic algorithm that takes as input* 1^λ*, and returns a public/secret key pair* $(pk, sk) \in \{0,1\}^p \times \{0,1\}^s$ *for some polynomials* $p = p(\lambda)$ *and* $s = s(\lambda)$*. The set of all secret keys (output by* $\mathsf{Gen}(1^\lambda)$*) and the (internal) randomness space of* Gen *are respectively denoted by* $\mathcal{SK}$ *and* $\mathcal{R}_g$*. (b)* Sign *is a probabilistic algorithm that takes as input a secret key* sk *and a message* $m \in \mathcal{M}$*, where* $\mathcal{M}$ *is the message space, and returns a signature* σ*. The (internal) randomness space of* Sign *is denoted by* $\mathcal{R}_s$*. (c)* Verify *is a deterministic algorithm that takes as input a public key* pk*, a message* m*, and a signature* σ*, and returns* 1 *(accept) or* 0 *(reject).*

A signature scheme is required to satisfy correctness, which means that $\mathsf{Verify}_{pk}(m, \sigma) = 1$ *holds for all* $\lambda \in \mathbb{N}$*, all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, all* $m \in \mathcal{M}$*, and all* $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$*.*

We now recall the definition of a unique signature scheme, in which the signing algorithm is deterministic and there exists only one valid signature for each pair of public key (not necessarily output by $\mathsf{Gen}(1^\lambda)$) and message.

Definition 8 (Unique signature [54]). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be a* unique signature scheme *if (a)* Sign *is deterministic, and (b) for all* $\lambda \in \mathbb{N}$*, all* $pk \in \{0,1\}^p$ *(possibly outside the support of* Gen*), and all* $m \in \mathcal{M}$*, there exists no pair* (σ, σ') *that simultaneously satisfies* $\sigma \neq \sigma'$ *and* $\mathsf{Verify}_{pk}(m, \sigma) = \mathsf{Verify}_{pk}(m, \sigma') = 1$*.*

One can easily see that a unique signature scheme can be cast as a PDP where Comp is the signing algorithm and Prove always returns the empty string.

We now define unique-key signatures, where there exists only one valid secret key for each public key (not necessarily output by $\mathsf{Gen}(1^\lambda)$).

Definition 9 (Unique-key signature). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be a* unique-key signature scheme *if there exists a deterministic PT algorithm* $\mathsf{UKCheck}$ *such that (a)* $\mathsf{UKCheck}(pk, sk) = 1$ *holds for all* $\lambda \in \mathbb{N}$ *and all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, and (b) for all* $\lambda \in \mathbb{N}$ *and all* $pk \in \{0,1\}^p$ *(possibly outside the support of* Gen*), there exists no pair* $(sk, sk') \in \{0,1\}^s \times \{0,1\}^s$ *that simultaneously satisfies* $sk \neq sk'$ *and* $\mathsf{UKCheck}(pk, sk) = \mathsf{UKCheck}(pk, sk') = 1$*.*

3.2 Security Notions for PDPs and Signatures

In this subsection, we define several security notions, called weak unpredictability, weak unforgeability, and weak tamper-resilience, for PDPs and signatures.

In [2], two security notions were defined for PDPs: unpredictability and pseudorandomness. A PDP satisfying the former (respectively, latter) security notion is a VUF (respectively, VRF). Moreover, unpredictability is weaker than pseudorandomness (see [2, Lemma 5]). In this paper, we define weak unpredictability, which is weaker than standard unpredictability. This security notion only guarantees that a PPT adversary, which is allowed to neither learn pk nor make any query, cannot output a valid input/output pair.[14] We also define a similar

[14] Since we aim at proving the impossibility on tamper-resilience of primitives, we would like to define their underlying security in a way as weak as possible.

security notion, called weak unforgeability, for signatures. Furthermore, we define two additional security notions, called statistical weak unpredictability and statistical weak unforgeability, which treat possibly inefficient adversaries. We also refer the reader to Section A in Supplementary Material for the standard security of unpredictability and unforgeability.

Definition 10 ((Statistical) weak unpredictability). *A PDP* $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ *is said to be* weakly unpredictable *(respectively,* statistically weakly unpredictable*), if for any PPT adversary (respectively, possibly inefficient adversary)* $\mathcal{A}$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), (x^*, y^*) \leftarrow \mathcal{A}(1^\lambda) : \mathsf{Comp}_{sk}(x^*) = y^*] \leq negl(\lambda).$$

Definition 11 ((Statistical) weak unforgeability). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be* weakly unforgeable *(respectively,* statistically weakly unforgeable*), if for any PPT adversary (respectively, possibly inefficient adversary)* $\mathcal{A}$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), (m^*, \sigma^*) \leftarrow \mathcal{A}(1^\lambda) : \mathsf{Verify}_{pk}(m^*, \sigma^*) = 1] \leq negl(\lambda).$$

One can easily see that weak unpredictability and weak unforgeability are much weaker than standard unpredictability and unforgeability notions, respectively. However, this does not straightforwardly mean that statistical weak unpredictability and statistical weak unforgeability are also weak since they treat possibly inefficient adversaries. Now we give two lemmas showing the equivalence between weak unpredictability and statistical weak unpredictability, and that between weak unforgeability and statistical weak unforgeability.

Lemma 1. *A PDP satisfies weak unpredictability (against non-uniform adversaries) if and only if it satisfies statistical weak unpredictability.*

Proof (of Lemma 1). Since it is straightforward that statistical weak unpredictability implies weak unpredictability, we focus on the opposite direction.

If a PDP $\Phi = (\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ is not statistically weakly unpredictable, then there exists a (possibly inefficient) adversary $\mathcal{A}$ that, on input a security parameter 1^λ, can output x^* and y^* such that $\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) : y^* = \mathsf{Comp}_{sk}(x^*)]$ is non-negligible. Therefore, we can hard-wire such (x^*, y^*) in a (non-uniform) PPT adversary $\mathcal{B}$, which can easily break the weak unpredictability of Φ by outputting (x^*, y^*), on input 1^λ. Hence, weak unpredictability implies statistical weak unpredictability. □

Lemma 2. *A signature scheme satisfies weak unforgeability (against non-uniform adversaries) if and only if it satisfies statistical weak unforgeability.*

We omit the proof of Lemma 2 since it is exactly the same except that we replace the winning condition $y^* = \mathsf{Comp}_{sk}(x^*)$ with $\mathsf{Verify}_{pk}(x^*, y^*) = 1$.

We now define weak tamper-resilience (WTR) for PDPs and signatures. An adversary against such security notions consists of three independent components

(**Tamper**, **Break**1, **Break**2). **Tamper** models a tampering function which determines a tampered secret key, on input a public/secret key pair. **Break**1 takes as input a public key and makes a computing or signing query. On receiving the answer generated by using the tampered secret key, **Break**2 tries to output a valid forgery. For simplicity, we allow the adversary to make only one tampering query and one signing query, and forbid the communication between **Break**1 and **Break**2. These security notions are (strictly) extremely weak versions of the unforgeability under chosen message and tampering attacks defined in [28,35] and there have already been several positive results satisfying them [24,23,28,35], except that we treat the plain model. As explained before, the more restrictive the adversary is, the stronger our negative results are. Here, notice that although we treat **Tamper** as a component of the adversary, it is essentially a tampering query made by the adversary since $(\mathbf{Break}_1, \mathbf{Break}_2)$ can neither learn its inputs nor communicate with it. The formal definitions are as follows.

Definition 12 (Weak tamper-resilient (WTR) PDP). *A PDP* $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ *is said to satisfy* WTR security*, if for any PPT adversary* $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$*, we have*

$$\begin{aligned}\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), sk' \leftarrow \mathbf{Tamper}(1^\lambda, pk, sk), x \leftarrow \mathbf{Break}1(1^\lambda, pk),\\ y = \mathsf{Comp}_{sk'}(x), (x^*, y^*) \leftarrow \mathbf{Break}2(1^\lambda, pk, y) :\\ x^* \neq x \wedge sk' \in \mathcal{SK} \wedge y^* = \mathsf{Comp}_{sk}(x^*)] \leq negl(\lambda).\end{aligned}$$

The winning condition that $sk' \in \mathcal{SK}$ follows [28]. It makes our results stronger since the adversary is more restricted.

Definition 13 (Weak tamper-resilient (WTR) signature). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to satisfy* WTR security*, if for any PPT adversary* $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$*, we have*

$$\begin{aligned}\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), sk' \leftarrow \mathbf{Tamper}(1^\lambda, pk, sk), m \leftarrow \mathbf{Break}1(1^\lambda, pk),\\ \sigma \leftarrow \mathsf{Sign}_{sk'}(m), (m^*, \sigma^*) \leftarrow \mathbf{Break}2(1^\lambda, pk, \sigma) :\\ m^* \neq m \wedge sk' \in \mathcal{SK} \wedge \mathsf{Verify}_{pk}(m^*, \sigma^*) = 1] \leq negl(\lambda).\end{aligned}$$

Notice that when showing impossibility of black-box reductions for proving WTR security, we need to consider reduction algorithms that can exploit an adversary $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$ in any way they want, i.e., adaptively make queries to **Tamper**, **Break**1, and **Break**2 in any order and for any times (on condition that the total number of queries is polynomial in λ). This is an obstacle we need to overcome in our formal proofs. Also, note that unlike previous works treating arbitrary tampering attacks [24,23,28,35], we define **Tamper** as part of the adversary. However, we do this only for simplicity and this does not make any additional restriction on the reductions we consider. We refer the reader to Section 1.3 for the remark on our model for a further discussion.

3.3 Simulatable Attacks for WTR Secure PDPs and Signatures

In this subsection, we focus on showing the existence of simulatable attacks on the WTR security of PDPs and unique signatures, which implies the impossibility of deriving their WTR security from any commonly used assumption via black-box reductions, due to Theorem 1.

Simulatable attack for WTR secure PDPs. We start with giving a theorem showing that if a PDP satisfies weak unpredictability, then there exists a simulatable attack on its WTR security.

Theorem 2. *For any weakly unpredictable PDP* $\Phi = (\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$*, there exists a simulatable attack on the WTR security of* Φ*.*

Overview and idea of the proof. Let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_p$, and (x, x^*) be elements in $\mathcal{X}$ such that $x \neq x^*$. We firstly construct an inefficient adversary $\mathcal{A}$, which is an ensemble of stateless algorithms $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break1}_{\lambda,f}, \mathbf{Break2}_{\lambda,f})\}_{\lambda\in\mathbb{N}, f\in\mathbb{F}_\lambda}$. Each $\mathcal{A}_{\lambda,f}$ hard-wires (x, x^*) and an element f in $\mathbb{F}_\lambda$. We show that each $\mathcal{A}_{\lambda,f}$ breaks the WTR security of Φ. Then we show the existence of a PPT stateful simulator **Sim** that can simulate the behavior of $\mathcal{A}_{\lambda,f}$, in the case that f is randomly chosen from $\mathbb{F}_\lambda$ (and hence is a random function with domain $\{0,1\}^p$ and range $\mathcal{R}_g \times \mathcal{R}_p$). We design $\mathcal{A}_{\lambda,f}$ as follows.

$\mathbf{Tamper}_{\lambda,f}(1^\lambda, pk, sk)$ runs $(r_g, r_p) = f(pk)$ and $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$ (where r_p is an internal random coin of Prove that is used when checking the validity of (pk, sk) which we will explain later). If (pk, sk) is valid, it outputs sk' as the tampered key. $\mathbf{Break1}_{\lambda,f}(pk)$ outputs x as the computing query. $\mathbf{Break2}_{\lambda,f}(pk, y)$ runs $(r_g, r_p) = f(pk)$ and $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$, and checks whether y is a correct answer for x (i.e., whether $y = \mathsf{Comp}_{sk'}(x)$). If the check works, it does an exhaustive search to find (y^*, π^*) such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$, and outputs (x^*, y^*) as a forgery. Otherwise, it aborts. One can see that $\mathcal{A}_{\lambda,f}$ breaks the WTR security of Φ due to correctness and uniqueness.

Next we explain why we can have a PPT stateful simulator **Sim** which is indistinguishable from $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) in the view of any (possibly inefficient) oracle-access machine $\mathcal{B}$. To make $\mathbf{Break2}_{\lambda,f}$ output a forgery (rather than abort), $\mathcal{B}$ has to answer the query x made by $\mathbf{Break1}_{\lambda,f}$. However, without having given $\mathbf{Tamper}_{\lambda,f}$ a valid key pair (pk, sk) previously, $\mathcal{B}$ would learn no information on (pk', sk'). The reason is that $\mathcal{B}$ only has black-box access to $\mathbf{Tamper}_{\lambda,f}$ and cannot see the structure of f. When $\mathbf{Tamper}_{\lambda,f}(pk, sk)$ aborts, no information on $f(pk)$ is revealed. In this case, $\mathcal{B}$ cannot answer x due to the weak unpredictability of Φ. Therefore, we can construct **Sim**, who simulates the outputs of f by executing lazy sampling (i.e., by randomly choosing (r_g, r_p) and keeping them in its internal list), and outputs a forgery by using a valid key sk having appeared in a **Tamper** query made by $\mathcal{B}$ previously.[15] If such sk does

[15] This is possible since **Sim** is stateful and can record the previously queried keys in its internal list.

not exist, **Sim** aborts. Due to uniqueness, the final outputs of $\mathbf{Break2}_{\lambda,f}$ and **Sim** must be identical, which guarantees that the interactions with $\mathcal{A}_{\lambda,f}$ and with **Sim** are indistinguishable in the view of $\mathcal{B}$.

More specifically, we show the indistinguishability by giving hybrid machines $\mathcal{A}_0$ and $\mathcal{A}_1$. $\mathcal{A}_0$ interacts with $\mathcal{B}$ in the same way as $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) does, except that it simulates the outputs of f by executing lazy sampling. $\mathcal{A}_1$ runs in the same way as $\mathcal{A}_0$ does except that it aborts if $\mathcal{B}$ makes a **Break**2 query (pk, y), where a valid secret key sk for pk has not appeared in a **Tamper** query previously. The indistinguishability between $\mathcal{A}_{\lambda,f}$ and $\mathcal{A}_0$ follows from the fact that the outputs of f look perfectly random in the view of $\mathcal{B}$. Furthermore, $\mathcal{A}_0$ is indistinguishable from $\mathcal{A}_1$, since without having given $\mathcal{A}_0$ a valid key pair (pk, sk) previously, all the **Break**2 queries including pk that $\mathcal{B}$ makes will lead $\mathcal{A}_0$ to abort. Otherwise, we can construct an adversary breaking the statistical weak unpredictability of Φ. However, since statistical weak unpredictability is equivalent to (computational) weak unpredictability (see Lemma 1), such an adversary cannot exist due to the weak unpredictability of Φ. At last, we need to show that $\mathcal{A}_1$ is indistinguishable from **Sim**. The only difference between them is that to output a forgery, $\mathcal{A}_1$ does an exhaustive search, while **Sim** exploits a valid secret key having appeared in a **Tamper** query previously. Due to the uniqueness of Φ, their forgeries (with respect to the same pk) must be identical, and hence $\mathcal{B}$ cannot distinguish the interactions with $\mathcal{A}_{\lambda,f}$ and with **Sim**.

Notice that we need to ensure that $\mathbf{Tamper}_{\lambda,f}$ and **Sim** output a tampered key only if they receive a valid key pair (pk, sk). Otherwise, **Sim** cannot simulate the behavior of $\mathcal{A}_{\lambda,f}$. Nevertheless, one may wonder how they check the validity, since there seems to be no checking algorithm for key pairs. To deal with this issue, we make them verify whether $\mathsf{Verify}_{pk}(x^*, \mathsf{Comp}_{sk}(x^*), \mathsf{Prove}_{sk}(x^*; r_p)) = 1$ instead. Although such a procedure does not check the validity of (pk, sk) directly, it guarantees that if (pk, sk) passes the verification, then $\mathbf{Break2}_{\lambda,f}$ can absolutely find a forgery for x^* with respect to pk by using its brute force, and **Sim** can output the same forgery by using sk.

Proof (of Theorem 2). For each $\lambda \in \mathbb{N}$, let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_p$, and x and x^* be arbitrary elements in $\mathcal{X}$ such that $x \neq x^*$. We define an inefficient class of stateless adversaries $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break1}_{\lambda,f}, \mathbf{Break2}_{\lambda,f})\}_{\lambda\in\mathbb{N}, f\in\mathbb{F}_\lambda}$, where x and x^* are hard-wired, as follows.

$\mathbf{Tamper}_{\lambda,f}(pk, sk)$:[16]
1. Compute $(r_g, r_p) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $\mathsf{Verify}_{pk}(x^*, \mathsf{Comp}_{sk}(x^*), \mathsf{Prove}_{sk}(x^*; r_p)) = 1$, return sk'. Otherwise, return $\perp$.

$\mathbf{Break1}_{\lambda,f}(pk)$:
1. Return x.

[16] For simplicity, we omit the procedure of length-checking since the security parameter is implicitly taken as input. The same argument is made for other algorithms.

Break$2_{\lambda,f}(pk, y)$:

1. Compute $(r_g, r_p) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $y = \mathsf{Comp}_{sk'}(x)$, then do an exhaustive search to find the lexicographically first pair (y^*, π^*) such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$, and return (x^*, y^*). If $y \neq \mathsf{Comp}_{sk'}(x)$ or such (y^*, π^*) does not exist, return $\bot$.

It is clear that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$ implies $y^* = \mathsf{Comp}_{sk}(x^*)$ if (pk, sk) is an honestly sampled key pair, due to the correctness and uniqueness of Φ. Furthermore, the checks done by $\mathbf{Tamper}_{\lambda,f}$ and **Break**$2_{\lambda,f}$ must work in a WTR security game due to correctness, and we have $x \neq x^*$. Hence, for each $f \in \mathbb{F}_\lambda$, $\mathcal{A}_{\lambda,f}$ breaks the WTR security of Φ with advantage 1.

We now define the PPT stateful simulator $\mathbf{Sim}(1^\lambda)$. It internally keeps a list Ł (which is initially empty) as a state, and answers queries as follows.

- On receiving a **Tamper** query (pk, sk):
  1. Search $(pk, \overline{sk}, r_g, r_p) \in$ Ł.[17] If the searching process failed (i.e., pk is not currently in Ł), randomly choose $(r_g, r_p) \leftarrow \mathcal{R}_g \times \mathcal{R}_p$, and add $(pk, \overline{sk}, r_g, r_p)$ (where $\overline{sk} = \bot$) to Ł.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $\mathsf{Verify}_{pk}(x^*, \mathsf{Comp}_{sk}(x^*), \mathsf{Prove}_{sk}(x^*; r_p)) = 1$, replace $\overline{sk}$ with sk in Ł (if $\overline{sk} = \bot$), and return sk'. Otherwise, return $\bot$.
- On receiving a **Break**1 query pk:
  1. Return x.
- On receiving a **Break**2 query (pk, y):
  1. Search $(pk, \overline{sk}, r_g, r_p) \in$ Ł. If the searching process failed or $\overline{sk} = \bot$, return $\bot$.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $y = \mathsf{Comp}_{sk'}(x)$, compute $y^* = \mathsf{Comp}_{\overline{sk}}(x^*)$ and return (x^*, y^*). Otherwise, return $\bot$.

Let $\mathcal{B}^{(\cdot)}$ be any (possibly inefficient) oracle-access probabilistic machine that makes at most polynomially many queries. We show that $|\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$ by giving hybrid machines.[18]

Hybrid machine $\mathcal{A}_0$: We define an inefficient *stateful* machine $\mathcal{A}_0$ as follows.

- On receiving a **Tamper** query (pk, sk):
  1. Search $(pk, \overline{sk}, r_g, r_p) \in$ Ł (where Ł is initialized with $\emptyset$). If the searching process failed, randomly choose $(r_g, r_p) \leftarrow \mathcal{R}_g \times \mathcal{R}_p$, and add $(pk, \overline{sk}, r_g, r_p)$ (where $\overline{sk} = \bot$) to Ł.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $\mathsf{Verify}_{pk}(x^*, \mathsf{Comp}_{sk}(x^*), \mathsf{Prove}_{sk}(x^*; r_p)) = 1$, replace $\overline{sk}$ with sk in Ł (if $\overline{sk} = \bot$), and return sk'. Otherwise, return $\bot$.
- On receiving a **Break**1 query pk:

[17] Searching processes succeed even if $\overline{sk}$ is $\bot$.

[18] Recall that during the execution of $\mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)$ or $\mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)$, $\mathcal{B}$ can adaptively make **Tamper**, **Break**1, and **Break**2 queries in any order and for any times (on condition that the total number of queries is polynomial in λ).

1. Return x.

– On receiving a **Break**2 query (pk, y):
1. Search $(pk, \overline{sk}, r_g, r_p) \in$ Ł. If the searching process failed, then randomly choose $(r_g, r_p) \leftarrow \mathcal{R}_g \times \mathcal{R}_p$, and add $(pk, \perp, r_g, r_p)$ to Ł.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $y = \mathsf{Comp}_{sk'}(x)$, then do an exhaustive search to find the lexicographically first pair (y^*, π^*) such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$, and return (x^*, y^*). If $y \neq \mathsf{Comp}_{sk'}(x)$ or such (y^*, π^*) does not exist, return $\perp$.

Lemma 3. $\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] = \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)]$.

Proof (of Lemma 3). Since f (used by $\mathcal{A}_{\lambda,f}$) is randomly sampled from $\mathbb{F}_\lambda$, the outputs of f are perfect randomness in the view of $\mathcal{B}$. Moreover, what $\mathcal{A}_0$ does is just simulating f by executing lazy sampling. Then this lemma follows from the fact that the views of $\mathcal{B}$ in the interactions with $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) and with $\mathcal{A}_0$ are identical. □

Hybrid machine $\mathcal{A}_1$: We now define an inefficient stateful machine $\mathcal{A}_1$ which is the same as $\mathcal{A}_0$, except that on receiving a **Break**2 query (pk, y), $\mathcal{A}_1$ aborts if there is no valid secret key $\overline{sk}$ for pk stored in Ł. In other words, $\mathcal{A}_1$ aborts if pk has not appeared in a **Tamper** query previously, the answer of which is not $\perp$. Formally, on receiving a **Break**2 query (pk, y), $\mathcal{A}_1$ runs as follows. (Bellow, the difference from $\mathcal{A}_0$ is only in Step 1, and is *emphasized*.)

1. Search $(pk, \overline{sk}, r_g, r_p) \in$ Ł. *If the searching process failed or $\overline{sk} = \perp$, return $\perp$.*
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $y = \mathsf{Comp}_{sk'}(x)$, do an exhaustive search to find the lexicographically first pair (y^*, π^*) such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$, and return (x^*, y^*). If $y \neq \mathsf{Comp}_{sk'}(x)$ or such (y^*, π^*) does not exist, return $\perp$.

Lemma 4. $|\Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$.

Proof (of Lemma 4). Let E be the event that during the execution of $\mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)$, $\mathcal{B}$ makes a **Break**2 query (pk, y) such that

– pk has not appeared in a valid **Tamper** query (the answer of which is not $\perp$) previously, and
– the answer to this **Break**2 query is *not* $\perp$.

If E does not occur during the execution of $\mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)$, then the view of $\mathcal{B}$ is identical to its view in the interaction with $\mathcal{A}_1$. Therefore, we have $|\Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)]| \leq \Pr[E]$. It remains to show that $\Pr[E]$ is negligible.

Claim 1 *Let the total number of (all kinds of) queries made by $\mathcal{B}$ be q. Then there exists an inefficient adversary $\mathcal{E}$ that breaks the statistical weak unpredictability of Φ with probability at least $\Pr[E]/q^2$.*

Proof idea of Claim 1. A tampered key is generated as $(pk', sk') \leftarrow \mathsf{Gen}(1^\lambda; r_g)$ where r_g is a randomness linked with a public key pk. If $\mathcal{B}$ does not make a valid **Tamper** query (which does not lead $\mathcal{A}_0$ to output $\perp$) including pk, it would learn no information on (pk', sk').[19] In this case, making a **Break**2 query, such that pk is included in this query but $\mathcal{A}_0$ does not return $\perp$, means forging a valid input/output pair under pk', due to the check done at the third step in the response to a **Break**2 query. Therefore, $\mathcal{B}$ can be used to break the statistical weak unpredictability of Φ. The formal proof is as follows.

Proof (of Claim 1). The description of $\mathcal{E}$ is as follows.

The challenger samples $(pk', sk') \leftarrow \mathsf{Gen}(1^\lambda)$ and gives 1^λ to $\mathcal{E}$. Then $\mathcal{E}$ randomly chooses $\hat{i}, \hat{j} \leftarrow \{1, \cdots, q\}$, and interacts with $\mathcal{B}$ in the same way as $\mathcal{A}_0$ does (during the execution of $\mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)$), except that

1. From the $\hat{i}$th query, every time when receiving a **Tamper** or **Break**2 query including pk used in the $\hat{i}$th query, $\mathcal{E}$ returns $\perp$ to $\mathcal{B}$ (except for the $\hat{j}$th query).[20]
2. On receiving the $\hat{j}$th query, if it is a **Break**2 query denoted by (pk, y), $\mathcal{E}$ returns (x, y) to the challenger and terminates. Otherwise $\mathcal{E}$ aborts.

Now we argue that $\mathcal{E}$ returns (x, y) such that $y = \mathsf{Comp}_{sk'}(x)$ with probability at least $\Pr[E]/q^2$.

During the execution of $\mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)$, when E occurs, $\mathcal{B}$ must have made a **Break**2 query (pk, y), such that (a) there exists no entry $(pk, \overline{sk}, r_g, r_p) \in \text{Ł}$ such that $\overline{sk} \neq \perp$, but (b) $\mathcal{A}_0$ does not return $\perp$ as its answer. Therefore, for randomly sampled $\hat{i}, \hat{j} \leftarrow \{1, \cdots, q\}$ (not learnt by $\mathcal{B}$), when E occurs, the probability that

- the $\hat{j}$th query is the first **Break**2 query satisfying the above two conditions (a) and (b), and
- the $\hat{j}$th query includes pk, which is firstly used in the $\hat{i}$th query,

is at least $1/q^2$. Since $\mathcal{E}$ perfectly simulates $\mathcal{A}_0$ in the interaction with $\mathcal{B}$ in this case (till the termination), it returns (x, y) such that $y = \mathsf{Comp}_{sk'}(x)$ with probability at least $\Pr[E]/q^2$. Notice that in this case, r_g linked with pk (used in the $\hat{j}$th query) is not used by $\mathcal{E}$, and we view (pk', sk') (generated by the challenger and not learnt by $\mathcal{E}$) as the key pair generated by using r_g. This completes the proof of Claim 1. □

Due to the equivalence between weak unpredictability and statistical weak unpredictability (see Lemma 1) and the assumption that Φ is weakly unpredictable, $\Pr[E]/q^2$ is negligible. Therefore, $\Pr[E]$ is negligible, completing the proof of Lemma 4. □

Lemma 5. $\Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)] = \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)]$.

Proof (of Lemma 5). The difference between **Sim** and $\mathcal{A}_1$ is that to return a correct input/output pair, **Sim** computes $y^* = \mathsf{Comp}_{\overline{sk}}(x^*)$ while $\mathcal{A}_1$ does

[19] As explained in the idea of the proof, since $\mathcal{B}$ only has black-box access to $\mathbf{Tamper}_{\lambda,f}$ and cannot see the structure of f, when $\mathbf{Tamper}_{\lambda,f}(pk, sk)$ aborts, no information on $f(pk)$ is revealed.

[20] $\mathcal{E}$ may terminate before receiving the $\hat{i}$th query.

an exhaustive search to find the lexicographically first pair (y^*, π^*) such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$. One can see that during the execution of $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$ and $\mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)$, $\mathsf{Verify}_{pk}(x^*, \mathsf{Comp}_{\overline{sk}}(x^*), \mathsf{Prove}_{\overline{sk}}(x^*; r_p)) = 1$ holds for all $(pk, \overline{sk}, r_g, r_p) \in \text{Ł}$ where $\overline{sk} \neq \perp$, due to the check done in the third step in the response to a **Tamper** query. Furthermore, there exists only one y^* such that $\mathsf{Verify}_{pk}(x^*, y^*, \pi^*) = 1$ for some π^*, due to the uniqueness of Φ. Therefore, the forgeries output by **Sim** and $\mathcal{A}$ with respect to the same pk are the same, i.e., the views of $\mathcal{B}$ in the interactions with **Sim** and with $\mathcal{A}_1$ are identical. □

Due to Lemmas 3 to 5, we have $|\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$, completing the proof of Theorem 2. □

Simulatable attack for WTR secure unique-key signatures. Next we give our negative result on tamper-resilient unique-key signatures.

Theorem 3. *For any weakly unforgeable unique-key signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*, there exists a simulatable attack on the WTR security of* Σ*.*

We refer the reader to Section B in Supplementary Material for the adversaries and simulator in this simulatable attack. Unlike those in the proof of Theorem 2, the adversaries and simulator in this simulatable attack check the validity of a key pair in a direct way by using the key checking algorithm $\mathsf{UKCheck}$, and the adversaries exhaustively search a secret key and forge a signature by using it. We omit the proof of the indistinguishability between the adversaries and simulator since it can be done by slightly modifying the proof of Theorem 2.

3.4 Summary of Negative Results on PDPs and Signatures

Since we have shown the existence of simulatable attacks on the WTR security of any weakly unpredictable PDP and any weakly unforgeable unique-key signature scheme, we obtain the following corollary, which is the first main result in our paper. It directly follows from Theorems 1, 2, and 3.

Corollary 1 *Let* Φ *(respectively,* Σ*) be a PDP (respectively, a unique-key signature scheme). If* Φ *(respectively,* Σ*) is weakly unpredictable (respectively, weakly unforgeable), then there exists no black-box reduction deriving the WTR security of* Φ *(respectively,* Σ*) from the security of any cryptographic game.*

This corollary implies the impossibility of deriving the (even extremely weak) tamper-resilience of PDPs and unique-key signatures from any common assumption, via black-box reductions.

As explained before, VRFs, VUFs, and unique signatures can be cast as PDPs. Furthermore, the security notions for VRFs and VUFs, called pseudorandomness and unpredictability respectively (see [2] for the formal definitions), are stronger than weak unpredictability, and a weakly unforgeable unique signature scheme can be viewed as a weakly unpredictable PDP. Hence, we obtain the following corollary derived from Corollary 1.

Corollary 2 *Let Φ be a VRF, a VUF, or a weakly unforgeable unique signature scheme. There exists no black-box reduction deriving the WTR security of Φ from the security of any cryptographic game.*

Negative result on injective OWFs. In [68], Wichs gave a negative result on leakage-resilient unique-key one-way relations. One may also wonder whether tamper-resilient injective OWFs are achievable. Indeed, Faonio and Venturi [28] implicitly used a tamper-resilient OWF, and we can follow them to combine a tamper-resilient injective OWF with the framework in [25] to obtain tamper-resilient unique-key signatures in the CRS model, where it is assumed that tamper-proof public parameters are available. However, a simplified version of the proof of Theorem 2 can be adopted to show the non-existence of black-box reductions deriving the tamper-resilience of OWFs from the security of any cryptographic game. We refer the reader to Section C in Supplementary Material for the definition, security notion, and simulatable attack for tamper-resilient injective OWFs. Notice that this negative result does not imply those on unique-key signatures and encryption schemes, since a public/secret key pair is not necessarily an output/input pair of an OWF, and a signature or a ciphertext does not necessarily include public keys.

Negative result on re-randomizable signatures. In [43], Hofheinz et al. extended the negative results on tight security of unique signatures, which was firstly proved by Coron [19] and then fixed by Kakvi and Kiltz [48], for the notion of re-randomizable signatures. This notion is more general and hence captures more instantiations, such as the Waters signature scheme [66] and its variant [43]. We argue that such an extension can also be adopted in our case. Namely, by slightly changing the proof of Theorem 2, we can prove the impossibility on re-randomizable signatures with tamper-resilience. Intuitively, like the uniqueness of a PDP, which guarantees that the outputs of the adversary and the simulator are the same, the re-randomizability guarantees that their (re-randomized) outputs are indistinguishable in the view of any (possibly inefficient) distinguisher. We refer the reader to Section D in Supplementary Material for the definition of re-randomizable signatures and the simulatable attack on their WTR security. We omit the formal proof since it can be done by slightly modifying the proof of Theorem 2.

We can also extend the negative result on unique-key signatures for signatures with key re-randomization [5], and the same extension can be adopted for the above mentioned result on injective OWFs and our result on PKE schemes introduced in the next section. These extensions are similar to the above one and hence we omit the details.

Remark on self-destructive and key-updating mechanisms. The tamper-resilience notions of many constructions are guaranteed by self-destructive or key-updating mechanism (e.g., [37,27,49,23,35]). The former allows a device to erase all internal data when tampering is detected, so that an adversary cannot obtain any information from further queries. The latter allows a device to update

its secret information. Since we treat adversaries that only make one tampering query and one computing query, our results (in both this section and the next section) capture tamper-resilient primitives with these two mechanisms as well.

4 Impossibility on Unique-Message PKE Schemes

In this section, we give a negative result on unique-message PKE schemes. Our result shows that if a unique-message PKE scheme satisfies some "extremely" weak security, then there exists no black-box reduction deriving its tamper-resilience in the plain model from any commonly used falsifiable assumption.

The rest of this section is organized as follows. In Section 4.1, we give the definition of a unique-message PKE scheme. In Section 4.2, we define several security notions. In Section 4.3, we show the existence of a weak simulatable attack on the tamper-resilience of unique-message PKE schemes. In Section 4.4, we summarize our negative result.

4.1 Definition of PKE Schemes

We now give the definition of a PKE scheme.

Definition 14 (Public key encryption (PKE)). *A PKE scheme consists of the PT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*. (a)* Gen *is a probabilistic algorithm that takes as input* 1^λ*, and returns a public/secret key pair* $(pk, sk) \in \{0,1\}^p \times \{0,1\}^s$ *for some polynomials* $p = p(\lambda)$ *and* $s = s(\lambda)$*. The set of all secret keys (output by* $\mathsf{Gen}(1^\lambda)$*) and the (internal) randomness space of* Gen *are denoted by* $\mathcal{SK}$ *and* $\mathcal{R}_g$*, respectively. (b)* Enc *is a probabilistic algorithm that takes as input a public key* pk *and a message* $m \in \mathcal{M}$*, where* $\mathcal{M}$ *is the message space, and returns a ciphertext* $ct \in \{0,1\}^c$ *for some polynomial* $c = c(\lambda)$*. The (internal) randomness space of* Enc *is denoted by* $\mathcal{R}_e$*. (c)* Dec *is a deterministic algorithm that takes as input a secret key* sk *and a ciphertext* ct*, and returns a message* $m \in \mathcal{M}$ *or* $\perp$*.*

A PKE scheme is required to satisfy correctness*, which means that* $\mathsf{Dec}_{sk}(ct) = m$ *holds for all* $\lambda \in \mathbb{N}$*, all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, all* $m \in \mathcal{M}$*, and all* $ct \leftarrow \mathsf{Enc}_{pk}(m)$*.*

We now define unique-message PKE schemes. For a unique-message PKE scheme, there exists a plaintext-recovering algorithm which can correctly decrypt a ciphertext, and it is required that when using it to decrypt a ciphertext (possibly outside the support of Enc), all valid secret keys with respect to a public key (possibly outside the support of Gen) should lead to the same decryption result.

Definition 15 (Unique-message PKE). *A PKE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be a* unique-message PKE scheme *if there exist two deterministic PT algorithm* $(\mathsf{UMCheck}, \mathsf{Rec})$ *such that (a)* $\mathsf{UMCheck}(pk, sk) = 1$ *holds for all* $\lambda \in \mathbb{N}$ *and all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, (b)* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Rec})$ *satisfies correctness, and (c) for all* $\lambda \in \mathbb{N}$*, all* $pk \in \{0,1\}^p$*, and all* $ct \in \{0,1\}^c$ *(possibly outside the support*

of Gen *and* Enc *respectively), there exists no pair* $(sk, sk') \in \{0,1\}^s \times \{0,1\}^s$ *that simultaneously satisfies* $\mathsf{UMCheck}(pk, sk) = \mathsf{UMCheck}(pk, sk') = 1$ *and* $\mathsf{Rec}_{sk}(ct) \neq \mathsf{Rec}_{sk'}(ct)$.

For ease of understanding the rationale behind the above definition, as an instance, we show how it captures the Cramer-Shoup scheme [20] in Section E.1 in Supplementary Material. Furthermore, we argue that this definition captures all the unique-key PKE schemes, which are defined in the same way as unique-key signatures, namely, there exists only one valid secret key for each public key (not necessarily output by $\mathsf{Gen}(1^\lambda)$). We refer the reader to Section E.2 in Supplementary Material for the formal definition and the proof that all unique-key PKE schemes fall under the definition of unique-message PKE schemes.

4.2 Security Notions for PKE Schemes

In this subsection, we define weak one-wayness and WTR security for PKE schemes.

At first, we give the definition of weak one-wayness. Such security only guarantees that an adversary, which is allow to neither learn pk nor make any decryption query, cannot recover a message from a randomly generated ciphertext.

Definition 16 (Weak one-wayness). *A PKE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be* weakly one-way*, if for any PPT adversary* $\mathcal{A}$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), m \leftarrow \mathcal{M}, ct \leftarrow \mathsf{Enc}_{pk}(m), \\ m' \leftarrow \mathcal{A}(1^\lambda, ct) : m = m'] \leq negl(\lambda).$$

Remark. We can also define statistical weak one-wayness by replacing "PPT adversary" with "(possibly inefficient) adversary". However, unlike the cases of unpredictability and unforgeability notions defined for PDPs and signatures, statistical weak one-wayness is *not necessarily* implied by weak one-wayness. Specifically, for a one-way PKE scheme, one can see that its security does not change if we put the public key into the ciphertext. The one-way security of the original PKE scheme implies the weak one-wayness of this modified scheme. However, if the public key is part of the ciphertext, the scheme cannot be statistically weakly one-way because an inefficient adversary can find a secret key corresponding to the public key and decrypt the challenge ciphertext.

We now define WTR security for PKE schemes. Like WTR security for signatures, an adversary in the security game consists of three independent components $(\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$. **Tamper** determines a tampered secret key, on input a public/secret key pair. **Break**1 takes as input a public key and makes a decryption query. On receiving the answer generated by using the tampered secret key and a challenge ciphertext, **Break**2 tries to decrypt the challenge ciphertext. Similarly to WTR security for signatures, we allow the adversary to make only one tampering query and one decryption query, and forbid the

communication between **Break**1 and **Break**2. If the message space is super-polynomially large, one may consider this security notion as an extremely weak version of the indistinguishability under chosen ciphertext attacks and tampering attacks defined in [28,35], except that we treat the plain model. As noted before, the more restrictive the adversary is, the stronger our negative result is. The formal definition is as follows.

Definition 17 (Weak tamper-resilient (WTR) PKE). *A PKE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to satisfy* WTR security*, if for any PPT adversary* $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break}1, \mathbf{Break}2)$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), sk' \leftarrow \mathbf{Tamper}(1^\lambda, pk, sk), ct \leftarrow \mathbf{Break}1(1^\lambda, pk), \\ m = \mathsf{Dec}_{sk'}(ct), m^* \leftarrow \mathcal{M}, ct^* \leftarrow \mathsf{Enc}_{pk}(m^*), m' \leftarrow \mathbf{Break}2(1^\lambda, pk, m, ct^*) : \\ sk' \in \mathcal{SK} \wedge m' = m^*] \leq negl(\lambda).$$

4.3 Weak Simulatable Attack for WTR Secure Unique-Message PKE Schemes

Next we give a theorem showing the existence of a weak simulatable attack on the WTR security of unique-message PKE schemes satisfying weak one-wayness. The basic idea of the proof is similar to that of Theorem 2. The main difference is that in this case, the indistinguishability between the inefficient adversary $\mathcal{A}$ and the PPT simulator **Sim** follows from the fact that without making a valid **Tamper** query (which does not lead $\mathcal{A}$ to output $\perp$), the distinguisher cannot answer a decryption query (rather than a signing query). Similarly to the simulatable attack for Theorem 3 (see Section B in Supplementary Material), $\mathcal{A}$ and **Sim** check the validity of a key pair in a direct way, by using the key checking algorithm $\mathsf{UMCheck}$.

Theorem 4. *For any weakly one-way unique-message PKE scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, there exists a weak simulatable attack on its WTR security.*

We give the overview and idea of the proof of Theorem 4 as below, and refer the reader to Section F in Supplementary Material for the full details.

Overview and idea of the proof of Theorem 4. Let $\mathsf{UMCheck}$ and Rec be the key checking algorithm and plaintext-recovering algorithm of Π, respectively, and $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$. We firstly construct an inefficient adversary $\mathcal{A}$, which is an ensemble of stateless algorithms $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break}1_{\lambda,f}, \mathbf{Break}2_{\lambda,f})\}_{\lambda\in\mathbb{N}, f\in\mathbb{F}_\lambda}$. Each $\mathcal{A}_{\lambda,f}$ hard-wires an element f in $\mathbb{F}_\lambda$. We show that each $\mathcal{A}_{\lambda,f}$ breaks the WTR security of Π, and there exists a PPT stateful simulator **Sim** that can simulate the behavior of $\mathcal{A}_{\lambda,f}$, in the case that f is chosen uniformly at random from $\mathbb{F}_\lambda$ (and hence is a random function with domain $\{0,1\}^p$ and range $\mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$). We design $\mathcal{A}_{\lambda,f}$ as follows.

$\mathbf{Tamper}_{\lambda,f}(1^\lambda, pk, sk)$ runs $(r_g, r_e, m) = f(pk)$ and $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$. If (pk, sk) is valid (i.e., $\mathsf{UMCheck}(pk, sk) = 1$), it outputs sk' as the tampered

key. $\mathbf{Break1}_{\lambda,f}(pk)$ runs $(r_g, r_e, m) = f(pk)$ and outputs $ct = \mathsf{Enc}_{pk'}(m; r_e)$ as the decryption query. $\mathbf{Break2}_{\lambda,f}(pk, m', ct^*)$ runs $(r_g, r_e, m) = f(pk)$ and $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$, and checks whether the answer of the decryption query is correct (i.e., whether $m' = m$). If the check works, it does an exhaustive search to find a secret key sk such that $\mathsf{UMCheck}(pk, sk) = 1$, and outputs $m^* = \mathsf{Rec}_{sk}(ct^*)$. Otherwise, it aborts. One can see that $\mathcal{A}_{\lambda,f}$ breaks the WTR security of Π due to the correctness of Π and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Rec})$, and the unique-message property of Π.

Next we explain why we can have a PPT stateful simulator **Sim** which is indistinguishable from $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) in the view of any oracle-access PPT machine $\mathcal{B}$. To make $\mathbf{Break2}_{\lambda,f}$ recover ct^* (rather than abort), $\mathcal{B}$ has to answer the query ct made by $\mathbf{Break1}_{\lambda,f}$. However, without having given $\mathbf{Tamper}_{\lambda,f}$ a valid key pair (pk, sk) previously, ct is just a randomly generated ciphertext in the view of $\mathcal{B}$, and $\mathcal{B}$ learns no information on (pk', sk'). The reason is that $\mathcal{B}$ only has black-box access to $\mathbf{Tamper}_{\lambda,f}$ and cannot see the structure of f. When $\mathbf{Tamper}_{\lambda,f}(pk, sk)$ aborts, no information on $f(pk)$ is revealed. In this case, $\mathcal{B}$ cannot answer ct due to the weak one-wayness of Π. Therefore, we can construct **Sim**, who simulates the outputs of f by executing lazy sampling, and decrypts ct^* by using a valid key having appeared in a **Tamper** query made by $\mathcal{B}$ previouly. If such a key does not exist, **Sim** aborts. Due to the unique-message property, the final outputs of $\mathbf{Break2}_{\lambda,f}$ and **Sim** must be identical, which guarantees that the interactions with $\mathcal{A}_{\lambda,f}$ and with **Sim** are indistinguishable in the view of $\mathcal{B}$.

More specifically, we show the indistinguishability by giving hybrid machines $\mathcal{A}_0$ and $\mathcal{A}_1$. $\mathcal{A}_0$ interacts with $\mathcal{B}$ in the same way as $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) does, except that it simulates the outputs of f by executing lazy sampling. $\mathcal{A}_1$ runs in the same way as $\mathcal{A}_0$ does except that on receiving a **Break**2 query (pk, m', ct^*) where m' is a correct answer for the decryption query, if a valid secret key sk for pk has appeared in a **Tamper** query previously, $\mathcal{A}_1$ directly computes $\mathsf{Rec}_{sk}(ct^*)$. Otherwise, it exhaustively searches sk and then computes $\mathsf{Rec}_{sk}(ct^*)$. The indistinguishability between $\mathcal{A}_{\lambda,f}$ and $\mathcal{A}_0$ follows from the fact that the outputs of f look perfectly random in the view of $\mathcal{B}$. Furthermore, due to the unique-message property of Π, the decryption results output by $\mathcal{A}_0$ and $\mathcal{A}_1$ (with respect to the same pk and ct^*) must be identical. Hence, $\mathcal{A}_0$ is indistinguishable from $\mathcal{A}_1$ in the view of $\mathcal{B}$. At last, we need to show that $\mathcal{A}_1$ is indistinguishable from **Sim**. The only difference between them is that when $\mathcal{B}$ makes a **Break**2 query (pk, m', ct^*), where a valid secret key sk for pk has not appeared in a **Tamper** query previously, $\mathcal{A}_1$ checks whether m' is a correct answer of the decryption query and exhaustively searches a secret key to decrypt ct^* (if the check works), while **Sim** just aborts. Then the indistinguishability between $\mathcal{A}_1$ and **Sim** follows from the fact that without having given $\mathcal{A}_1$ a valid key pair (pk, sk) previously, all the **Break**2 queries including pk that $\mathcal{B}$ makes will lead $\mathcal{A}_1$ to abort. Otherwise, we can construct an adversary breaking the weak one-wayness of Π.

Remark. By slightly modifying the above proof, we can also prove that if a unique-message PKE scheme satisfies statistical weak one-wayness, then there exists a simulatable attack on its WTR security.

Furthermore, it is clear that for any honestly generated ciphertext, any honestly generated secret key should lead to the same decryption result, due to correctness. Therefore, we can slightly modify the proof of Theorem 4 to show that for a weakly one-way (respectively, statistically weakly one-way) PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, if there exists an algorithm that can efficiently check whether a secret key and a ciphertext are in the support of Gen and Enc (with respect to a public key) respectively, then there exists a weak simulatable attack (respectively, (standard) simuatable attack) on its WTR security.

4.4 Summary of a Negative Result on Unique-Message PKE

Since we have shown the existence of a weak simulatable attack on the WTR security of any weakly one-way unique-message PKE schemes in the last subsection, we obtain the following corollary, which is the second main result in our paper. It follows directly from Theorems 1 and 4.

Corollary 3 *Let Π be a unique-message PKE scheme. If Π is weakly one-way, then there exists no black-box reduction deriving the WTR security of Π from the security of any restricted cryptographic game.*

This result implies the impossibility of deriving the (even extremely weak) tamper-resilience of a large class of PKE schemes from any common falsifiable assumption, via black-box reductions.

5 Open Problems

Primitives circumventing our results. Our impossibility results capture a wide class of tamper-resilient cryptographic primitives in the plain model, while it remains open whether achieving ones (against arbitrary tampering attacks) outside of the class is possible. One may try to circumvent our results by giving a tamper-resilient PKE scheme, which is weakly one-way but not statistically weakly one-way, under non-falsifiable assumptions. Furthermore, it might be promising to combine such a PKE scheme with an NIZK proof system against untrusted CRSs [9] to construct a tamper-resilient signature scheme, under the framework proposed by Dodis et al. [25].

Impossibility on primitives in the CRS model. One may also wonder whether it is possible to construct primitives captured by our negative results in the CRS model (where the existence of tamper-proof public parameters is assumed). If we strengthen the notions of statistical weak unpredictability and unforgeability, and weak one-wayness by allowing the public parameter to be a maliciously generated one (rather than an honestly sampled one), then our impossibility results also hold in this model. However, it is apparent that such security notions

are not implied by standard ones. Studying PKE schemes satisfying standard security but not satisfying ones against maliciously generated public parameters might serve as a starting point.

Acknowledgement

A part of this work was supported by the National Natural Science Foundation for Young Scientists of China under Grant Number 62002049, the Fundamental Research Funds for the Central Universities under Grant Numbers ZYGX2020J017, ZYGX2020ZB020, ZYGX2020ZB019, the Sichuan Science and Technology Program under Grant Numbers 2019YFG0506, 2020YFG0292, 2019YFG0505, 2020YFG0460, 2020YFG0462, Input Output Cryptocurrency Collaborative Research Chair funded by IOHK, JST OPERA JPMJOP1612, JST CREST JPMJCR14D6, JSPS KAKENHI JP16H01705, JP17H01695.

References

1. B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. A subversion-resistant SNARK. In *ASIACRYPT 2017*.
2. M. Abe, J. Camenisch, R. Dowsley, and M. Dubovitskaya. On the impossibility of structure-preserving deterministic primitives. In *TCC 2014*.
3. M. Abe, J. Groth, and M. Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In *ASIACRYPT 2011*.
4. G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. In *ACM CCS 2015*.
5. C. Bader, T. Jager, Y. Li, and S. Schäge. On the impossibility of tight cryptographic reductions. In *EUROCRYPT 2016*.
6. M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *EUROCRYPT 2016*.
7. M. Bellare and D. Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO 2010*.
8. M. Bellare, D. Cash, and R. Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT 2011*.
9. M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: security in the face of parameter subversion. In *ASIACRYPT 2016*.
10. M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and Applications. In *EUROCRYPT 2003*.
11. M. Bellare, K. G. Paterson, and S. Thomson. RKA security beyond the linear barrier: IBE, Encryption and Signatures. In *ASIACRYPT 2012*.
12. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO 1997*.
13. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2).
14. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *EUROCRYPT 1998*.
15. X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM CCS 2005*.

16. Z. Brakerski and N. Döttling. Hardness of LWE on general entropic distributions. In *EUROCRYPT 2020*.
17. S. Chakraborty, M. Prabhakaran, and D. Wichs. Witness maps and applications. In *PKC 2020*.
18. Y. Chen, B. Qin, J. Zhang, Y. Deng, and S. S. M. Chow. Non-malleable functions and their applications. In *PKC 2016*.
19. J. Coron. Optimal security proofs for PSS and other signature schemes. In *EUROCRYPT 2002*.
20. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1).
21. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3).
22. D. Dachman-Soled and M. Kulkarni. Upper and lower bounds for continuous non-malleable codes. In *PKC 2019*.
23. I. Damgård, S. Faust, P. Mukherjee, and D. Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. *J. Cryptology*, 30(1).
24. I. Damgård, S. Faust, P. Mukherjee, and D. Venturi. The chaining lemma and its application. In *ICITS 2015*.
25. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT 2010*.
26. Y. Dodis, V. Vaikuntanathan, and D. Wichs. Extracting randomness from extractor-dependent sources. In *EUROCRYPT 2020*.
27. S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. In *ICS 2010*.
28. A. Faonio and D. Venturi. Efficient public-key cryptography with bounded leakage and tamper resilience. In *ASIACRYPT 2016*.
29. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. Continuous non-malleable codes. In *TCC 2014*.
30. S. Faust, P. Mukherjee, D. Venturi, and D. Wichs. Efficient non-malleable codes and key derivation for poly-size tampering circuits. *IEEE Trans. Information Theory*, 62(12).
31. M. Fischlin, P. Harasser, and C. Janson. Signatures from sequential-or proofs. In *EUROCRYPT 2020*.
32. M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT 2010*.
33. G. Fuchsbauer. Subversion-zero-knowledge SNARKs. In *PKC 2018*.
34. E. Fujisaki and K. Xagawa. Efficient RKA-secure KEM and IBE schemes against invertible functions. In *LATINCRYPT 2015*.
35. E. Fujisaki and K. Xagawa. Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions. In *ASIACRYPT 2016*.
36. T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4).
37. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC 2004*.
38. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC 2011*.
39. S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *CRYPTO 1992*.
40. I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In *TCC 2009*.

41. K. Haralambiev, T. Jager, E. Kiltz, and V. Shoup. Simple and efficient public-key encryption from computational Diffie-Hellman in the standard model. In *PKC 2010*.
42. J. Hesse, D. Hofheinz, and L. Kohl. On tightly secure non-interactive key exchange. In *CRYPTO 2018*.
43. D. Hofheinz, T. Jager, and E. Knapp. Waters signatures with optimal security reduction. In *PKC 2012*.
44. S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. In *EUROCRYPT 2010*.
45. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO 2009*.
46. Z. Jafargholi and D. Wichs. Tamper detection and continuous non-malleable codes. In *TCC 2015*.
47. S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT 2004*.
48. S. A. Kakvi and E. Kiltz. Optimal security proofs for full domain hash, revisited. In *EUROCRYPT 2012*.
49. Y. T. Kalai, B. Kanukurthi, and A. Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO 2011*.
50. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO 2016*.
51. D. W. Kravitz. Digital signature algorithm. US Patent 5,231,668.
52. M. Liskov. Updatable zero-knowledge databases. In *ASIACRYPT 2005*.
53. F. Liu and A. Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO 2012*.
54. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO 2002*.
55. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *FOCS 1999*.
56. S. Micali and L. Reyzin. Soundness in the public-key model. In *CRYPTO 2001*.
57. S. Micali and R. L. Rivest. Micropayments revisited. In *CT-RSA 2002*.
58. A. Morgan and R. Pass. On the security loss of unique signatures. In *TCC 2018*.
59. M. Naor. On cryptographic assumptions and challenges. In *CRYPTO 2003*.
60. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT 2005*.
61. R. Pass. Limits of provable security from standard assumptions. In *STOC 2011*.
62. B. Qin, S. Liu, T. H. Yuen, R. H. Deng, and K. Chen. Continuous non-malleable key derivation and its application to related-key security. In *PKC 2015*.
63. T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In *EUROCRYPT 2011*.
64. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1).
65. Y. Wang, T. Matsuda, G. Hanaoka, and K. Tanaka. Memory lower bounds of reductions revisited. In *EUROCRYPT (1)*.
66. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*.
67. H. Wee. Public key encryption against related key attacks. In *PKC 2012*.
68. D. Wichs. Barriers in cryptography with weak, correlated and leaky sources. In *ITCS 2013*.
69. J. Zhang, Z. Zhang, Y. Chen, Y. Guo, and Z. Zhang. Black-box separations for one-more (static) CDH and its generalization. In *ASIACRYPT 2014*.

Supplementary Material

A Standard Definitions

In this section, we recall the standard definitions of unpredictability and unforgeability.

Definition 18 (Unpredictability). *A PDP* $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ *is said to be* unpredictable *if for any PPT adversary* $\mathcal{A}$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), (x^*, y^*) \leftarrow \mathcal{A}^{\mathbf{CompO}(\cdot)}(1^\lambda, pk) : x^* \notin \mathcal{Q}_x \wedge \mathsf{Comp}_{sk}(x^*) = y^*] \leq negl(\lambda),$$

where $\mathbf{CompO}(\cdot)$ *on input* x *returns* $y = \mathsf{Comp}_{sk}(x)$ *to* $\mathcal{A}$ *and adds* x *to* $\mathcal{Q}_x$ *(initialized as* $\emptyset$*).*

Definition 19 (Unforgeability). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be* unforgeable*, if for any PPT adversary* $\mathcal{A}$*, we have*

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathbf{SignO}(\cdot)}(1^\lambda, pk) : m^* \notin \mathcal{Q}_m \wedge \mathsf{Verify}_{pk}(m^*, \sigma^*) = 1] \leq negl(\lambda).$$

where $\mathbf{SignO}(\cdot)$ *on input* m *returns* $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ *to* $\mathcal{A}$ *and adds* m *to* $\mathcal{Q}_m$ *(initialized as* $\emptyset$*).*

B Simulatable Attack for Theorem 3

In this section, we show the constructions of the (inefficient) adversaries and (PPT) simulator in the simulatable attack for Theorem 3.

Inefficient adversaries. Let $\mathsf{UKCheck}$ be the key checking algorithm of Σ. For each $\lambda \in \mathbb{N}$, let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_s$, and m and m^* be arbitrary elements in $\mathcal{M}$ such that $m \neq m^*$. We define the inefficient class of stateless adversaries $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break1}_{\lambda,f}, \mathbf{Break2}_{\lambda,f})\}_{\lambda \in \mathbb{N}, f \in \mathbb{F}_\lambda}$, where m and m^* are hard-wired, as follows.

$\mathbf{Tamper}_{\lambda,f}(pk, sk)$:
1. If $\mathsf{UKCheck}(pk, sk) \neq 1$, return $\perp$.
2. Compute $(r_g, r_s) = f(pk)$.
3. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
4. Return sk'.

$\mathbf{Break1}_{\lambda,f}(pk)$:
1. Return m.

$\mathbf{Break2}_{\lambda,f}(pk, \sigma)$:
1. Compute $(r_g, r_s) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.

3. If $\mathsf{Verify}_{pk'}(m,\sigma) = 1$, then do an exhaustive search to find sk such that $\mathsf{UKCheck}(pk, sk) = 1$, compute $\sigma^* = \mathsf{Sign}_{sk}(m^*; r_s)$, and return (m^*, σ^*). If $\mathsf{Verify}_{pk'}(m,\sigma) \neq 1$ or such sk does not exist, return $\perp$.

It is straightforward that for each $f \in \mathbb{F}_\lambda$, $\mathcal{A}_{\lambda,f}$ breaks the WTR security with advantage 1, due to the correctness and the unique-key property of Σ.

Simulator. We now define the PPT stateful simulator $\mathbf{Sim}(1^\lambda)$. It internally keeps a list Ł (which is initially empty) as a state, and answers queries as follows.

- On receiving a **Tamper** query (pk, sk):
  1. If $\mathsf{UKCheck}(pk, sk) \neq 1$, return $\perp$.
  2. Search $(pk, \overline{sk}, r_g, r_s) \in$ Ł. If the searching process failed (i.e., pk is not currently in Ł), randomly choose $(r_g, r_s) \leftarrow \mathcal{R}_g \times \mathcal{R}_s$, and add (pk, sk, r_g, r_s) to Ł.
  3. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  4. Return sk'.
- On receiving a **Break**1 query pk:
  1. Return m.
- On receiving a **Break**2 query (pk, σ):
  1. Search $(pk, \overline{sk}, r_g, r_s) \in$ Ł. If the searching process failed, return $\perp$.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $\mathsf{Verify}_{pk'}(m,\sigma) = 1$, compute $\sigma^* = \mathsf{Sign}_{\overline{sk}}(m^*; r_s)$ and return (m^*, σ^*). Otherwise, return $\perp$.

As explained in the main body of this paper, the proof of the indistinguishability between the interactions with $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) and with **Sim**, which is based on the weak unforgeability and unique-key property of Σ, is similar to that in the proof of Theorem 2.

Remark on more restricted tampering attacks. We can also extend the above simulatable attack to one with respect to more restricted tampering attacks, which only allow **Tamper** to take sk as input.[21] Since security treating such attacks is weaker than WTR security, this extension helps us achieve stronger negative results. The same argument is also made for the simulatable attack for injective one-way functions in Section C in Supplementary Material. We omit the details since the extensions are straightforward.

C Impossibility on Injective One-Way Functions

In this section, we give a negative result on tamper-resilient injective OWFs. Our result shows that for an injective OWF, there exists no black-box reduction deriving its tamper-resilience in the plain model from any commonly used assumption.

[21] In this case, (r_g, r_s) is computed as $(r_g, r_s) = f(sk)$ and **Break**2 exhaustively searches sk at the first step. The adversary can still be simulated due to the unique-key property.

C.1 Definition of WTR Secure Injective One-Way Functions

We now define WTR secure injective OWFs. An adversary against their security consists of two independent components $(\mathbf{Tamper}, \mathbf{Break})$. **Tamper** determines a tampered input value, on input an input/output pair of the function. **Break** takes as input both a challenge instance and the output for a tampered input, and tries to recover the preimage. Similarly to the cases of signatures and PKE schemes, we allow the adversary to make only one tampering query.

Definition 20 (Weak tamper-resilient injective OWF). *A function* $\mathsf{F} : \{0,1\}^s \to \{0,1\}^p$*, where* $s = s(\lambda)$ *and* $p = p(\lambda)$ *are some polynomials, is said to be a WTR secure injective OWF if (a) there exists no pair* $(x, x') \in \{0,1\}^s \times \{0,1\}^s$ *that simultaneously satisfies* $x \neq x'$ *and* $\mathsf{F}(x) = \mathsf{F}(x')$*, and (b) for any PPT adversary* $\mathcal{A} = (\mathbf{Tamper}, \mathbf{Break})$*, we have*

$$\Pr[x \leftarrow \{0,1\}^s, x' \leftarrow \mathbf{Tamper}(1^\lambda, \mathsf{F}(x), x) : x \leftarrow \mathbf{Break}(1^\lambda, \mathsf{F}(x), \mathsf{F}(x'))] \leq negl(\lambda).$$

C.2 Simulatable Attacks for WTR Secure Injective OWFs

In this subsection, we give a negative result on WTR secure injective OWFs.

Theorem 5. *For any WTR secure injective OWF* $\mathsf{F} : \{0,1\}^s \to \{0,1\}^p$*, there exists a simulatable attack on the security of* F*.*

Inefficient adversaries. For each $\lambda \in \mathbb{N}$, let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \{0,1\}^s$. We define an inefficient class of stateless adversaries $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break}_{\lambda,f})\}_{\lambda\in\mathbb{N}, f\in\mathbb{F}_\lambda}$ as follows.

$\mathbf{Tamper}_{\lambda,f}(y, x)$:
1. If $\mathsf{F}(x) = y$, return $x' = f(y)$. Otherwise, return $\perp$.

$\mathbf{Break}_{\lambda,f}(y, y')$:
1. Compute $x' = f(y)$.
2. If $y' = \mathsf{F}(x')$, then do an exhaustive search to find x^* such that $\mathsf{F}(x^*) = y$, and return x^*. If $y' \neq \mathsf{F}(x')$ or such x^* does not exist, return $\perp$.

It is clear that for each $f \in \mathbb{F}_\lambda$, $\mathcal{A}_{\lambda,f}$ breaks the WTR security of F with advantage 1.

Simulator. We now define the PPT stateful simulator $\mathbf{Sim}(1^\lambda)$. It internally keeps a list Ł (which is initially empty) as a state, and answers queries as follows.

- On receiving a **Tamper** query (y, x):
  1. If $\mathsf{F}(x) = y$, search $(y, \overline{x}, x') \in$ Ł. If the searching process failed (i.e., y is not currently in Ł), randomly choose $x' \leftarrow \{0,1\}^s$, add (y, x, x') to Ł, and return x'. If $\mathsf{F}(x) \neq y$, return $\perp$.
- On receiving a **Break** query (y, y'):
  1. Search $(y, \overline{x}, x') \in$ Ł. If the searching process failed, return $\perp$.
  2. If $y' = \mathsf{F}(x')$, return $\overline{x}$. Otherwise, return $\perp$.

The proof of the indistinguishability between the interactions with $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) and with **Sim** is based on the fact that without having given $\mathbf{Tamper}_{\lambda,f}$ a valid output/input pair (y, x) previously, $y' = \mathsf{F}(x') = \mathsf{F}(f(y))$ is information theoretically hidden (since F is injective), which implies that all the **Break** queries including y will lead $\mathbf{Break}_{\lambda,f}$ to abort. Since the proof is just a simplified version of the proofs of Theorems 2 and 3, we omit the full details.

Following from Theorems 1 and 5, we summarize our negative result on injective OWFs as the following corollary.

Corollary 4 *There exists no black-box reduction deriving the security of a WTR secure injective OWF from the security of any cryptographic game.*

D Impossibility on Re-Randomizable Signatures

In this section, we give the definition of re-randomizable signatures and the simulatable attack on their WTR secuirty.

D.1 Definition of Re-Randomizable Signatures

We now recall the definition of a re-randomizable signature scheme, in which signatures are efficiently re-randomizable.

Definition 21 (Re-randomizable signature [43]). *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be* re-randomizable *if there exists a PPT algorithm* ReRand *such that* $\sigma' \leftarrow \mathsf{ReRand}(pk, m, \sigma)$ *is distributed uniformly over* $\mathcal{S}(pk, m)$ *for all* $\lambda \in \mathbb{N}$*, all* $pk \in \{0,1\}^p$ *(possibly outside the support of* Gen*), all* $m \in \mathcal{M}$*, and all* σ *such that* $\mathsf{Verify}(pk, m, \sigma) = 1$*. Here,* $\mathcal{S}(pk, m)$ *denotes the set of all* σ *such that* $\mathsf{Verify}(pk, m, \sigma) = 1$*, and the randomness space of* ReRand *is denoted by* $\mathcal{R}_r$*.*

One can see that unique signatures can be treated as special cases of re-randomizable ones, where Sign is deterministic, and $|\mathcal{S}(pk, m)| \leq 1$ holds for all $\lambda \in \mathbb{N}$, all $pk \in \{0,1\}^p$, and all $m \in \mathcal{M}$.

D.2 Simulatable Attack for WTR Secure Re-Randomizable Signatures

We now give a negative result on re-randomizable signatures.

Theorem 6. *For any weakly unforgeable re-randomizable signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*, there exists a simulatable attack on the WTR security of* Σ*.*

The constructions of the (inefficient) adversaries and (PPT) simulator in the simulatable attack for Theorem 6 are as follows.

Inefficient adversaries. Let ReRand be the re-randomization algorithm of Σ. For each $\lambda \in \mathbb{N}$, let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_s \times \mathcal{R}_r$, and m and m^* be arbitrary elements in $\mathcal{M}$ such that $m \neq m^*$. We define an inefficient class of stateless adversaries $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break1}_{\lambda,f}, \mathbf{Break2}_{\lambda,f})\}_{\lambda\in\mathbb{N},f\in\mathbb{F}_\lambda}$, where m and m^* are hard-wired, as follows.

$\mathbf{Tamper}_{\lambda,f}(pk, sk)$:
1. Compute $(r_g, r_s, r_r) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $\mathsf{Verify}_{pk}(m^*, \mathsf{Sign}_{sk}(m^*; r_s)) = 1$, return sk'. Otherwise, return $\perp$.

$\mathbf{Break1}_{\lambda,f}(pk)$:
1. Return m.

$\mathbf{Break2}_{\lambda,f}(pk, \sigma)$:
1. Compute $(r_g, r_s, r_r) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. If $\mathsf{Verify}_{pk'}(m, \sigma) = 1$, then do an exhaustive search to find the lexicographically first σ^* such that $\mathsf{Verify}_{pk}(m^*, \sigma^*) = 1$, compute $\sigma_r^* = \mathsf{ReRand}(pk, m^*, \sigma^*; r_r)$, and return (m^*, σ_r^*). If $\mathsf{Verify}_{pk'}(m, \sigma) \neq 1$ or such (m^*, σ^*) does not exist, return $\perp$.

It is straightforward that for each $f \in \mathbb{F}_\lambda$, $\mathcal{A}_{\lambda,f}$ breaks the WTR security with advantage 1, due to the correctness and re-randomizability of Σ.

Simulator. We now define the PPT stateful simulator $\mathbf{Sim}(1^\lambda)$. It internally keeps a list Ł (which is initially empty) as a state, and answers queries as follows.

- On receiving a **Tamper** query (pk, sk):
  1. Search $(pk, \overline{sk}, r_g, r_s, r_r) \in$ Ł. If the searching process failed (i.e., pk is not currently in Ł), randomly choose $(r_g, r_s, r_r) \leftarrow \mathcal{R}_g \times \mathcal{R}_s \times \mathcal{R}_r$, and add $(pk, \overline{sk}, r_g, r_s, r_r)$ (where $\overline{sk} = \perp$) to Ł.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $\mathsf{Verify}_{pk}(m^*, \mathsf{Sign}_{sk}(m^*; r_s)) = 1$, replace $\overline{sk}$ with sk in Ł (if $\overline{sk} = \perp$), and return sk'. Otherwise, return $\perp$.
- On receiving a **Break**1 query pk:
  1. Return m.
- On receiving a **Break**2 query (pk, σ):
  1. Search $(pk, \overline{sk}, r_g, r_s, r_r) \in$ Ł. If the searching process failed or $\overline{sk} = \perp$, return $\perp$.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. If $\mathsf{Verify}_{pk'}(m, \sigma) = 1$, compute $\sigma^* = \mathsf{Sign}_{\overline{sk}}(m^*; r_s)$, $\sigma_r^* = \mathsf{ReRand}(pk, m^*, \sigma^*; r_r)$, and return (m^*, σ_r^*). Otherwise, return $\perp$.

As explained in the main body of this paper, the proof of the indistinguishability between the interactions with $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) and with **Sim**, which is based on the weak unforgeability and re-randomizability of Σ, is similar to that in the proof of Theorem 2.

Following from Theorems 1 and 6, we summarize our negative result on re-randomizable signatures as the following corollary.

Corollary 5 *Let Σ be a re-randomizable signature scheme. If Σ is weakly unforgeable, then there exists no black-box reduction deriving the WTR security of Σ from the security of any cryptographic game.*

E Instances of Unique-Message PKE

In this section, we give examples of unique-message PKE schemes. Specifically, we show that the Cramer-Shoup scheme [20] and all the unique-key PKE schemes fall under the definition of unique-message PKE schemes.

E.1 The Cramer-Shoup Scheme

In this subsection, we show that the Cramer-Shoup scheme is a unique-message PKE scheme.

Let λ be the security parameter, $\mathbb{G}$ be a cyclic group of prime order p, and $\{H_k : \mathbb{G}^3 \to \mathbb{Z}_p\}_{k \in K}$ be a keyed collision resistant hash function. The Cramer-Shoup scheme is as follows.

- $\mathsf{Gen}(1^\lambda)$:
  1. Randomly choose $(g_1, g_2) \leftarrow \mathbb{G}^2$ and $k \leftarrow K$.
  2. Randomly choose $(x_1, x_2, y_1, y_2, z) \leftarrow \mathbb{Z}_p^5$.
  3. Compute $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$.
  4. Return (pk, sk), where $pk = (k, c, d, h, g_1, g_2)$ and $sk = (k, x_1, x_2, y_1, y_2, z, g_1, g_2)$.
- $\mathsf{Enc}_{pk}(m)$:
  1. Parse $pk = (k, c, d, h, g_1, g_2)$.
  2. Randomly choose $r \leftarrow \mathbb{Z}_p$.
  3. Compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $\alpha = H_k(u_1, u_2, e)$, and $v = c^r d^{r\alpha}$.
  4. Return $ct = (u_1, u_2, e, v)$.
- $\mathsf{Dec}_{sk}(ct)$:
  1. Parse $sk = (k, x_1, x_2, y_1, y_2, z, g_1, g_2)$ and $ct = (u_1, u_2, e, v)$.
  2. Compute $\alpha = H_k(u_1, u_2, e)$.
  3. If $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha \neq v$, return $\perp$.
  4. Return $m = e/(u_1^z)$.

One can easily verify the following lemma. We give its proof for completeness.

Lemma 6. *The Cramer-Shoup scheme is a unique-message PKE scheme.*

Proof. Let $\mathsf{UMCheck}$ be the checking algorithm, which takes as input $pk = (k, c, d, h, g_1, g_2)$ and $sk = (k, x_1, x_2, y_1, y_2, z, g_1, g_2)$, and checks whether $(h, g_1) \in \mathbb{G}^2$, $z \in \mathbb{Z}_p$, and $h = g_1^z$. If the check works, it outputs 1. Otherwise, it outputs 0. It is obvious that $\mathsf{UMCheck}(pk, sk) = 1$ holds for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.

Let Rec be a plaintext-recovering algorithm which takes as input $sk = (k, x_1, x_2, y_1, y_2, z, g_1, g_2)$ and $ct = (u_1, u_2, e, v)$, and checks whether $(e, u_1) \in \mathbb{G}^2$. If the check works, Rec outputs $m = e/(u_1^z)$. Otherwise, it outputs $\perp$. Since Rec is the same as Dec, except that it additionally checks whether $(e, u_1) \in \mathbb{G}^2$, and

skips checking validity of ciphertexts (i.e., Steps 2 and 3 in Dec), $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Rec})$ satisfies correctness, due to the correctness of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Furthermore, for all $pk = (k, c, d, h, g_1, g_2)$ and $sk = (k, x_1, x_2, y_1, y_2, z, g_1, g_2)$ such that $\mathsf{UMCheck}(pk, sk) = 1$, we have $h = g_1^z$, and there exists no (z, z') such that $h = g_1^z = g_1^{z'}$ and $e/(u_1^z) \neq e/(u_1^{z'})$. Hence, for any ct and pk, there exists no pair (sk, sk') that simultaneously satisfies $\mathsf{UMCheck}(pk, sk) = \mathsf{UMCheck}(pk, sk') = 1$ and $\mathsf{Rec}_{sk}(ct) \neq \mathsf{Rec}_{sk'}(ct)$. This completes the proof. □

E.2 Unique-Key PKE Scheme

In this subsection, we give the definition of a unique-key PKE scheme, which captures many existing implementations (e.g., [64,36,15,41]). Then we show that any unique-key PKE scheme can be viewed as a unique-message PKE scheme.

Definition 22 (Unique-key PKE). *A PKE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be a* unique-key PKE scheme*, if there exists a deterministic PT algorithm* $\mathsf{UKCheck}$ *such that (a)* $\mathsf{UKCheck}(pk, sk) = 1$ *holds for all* $\lambda \in \mathbb{N}$ *and all* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, and (b) for all* $\lambda \in \mathbb{N}$ *and all* $pk \in \{0,1\}^p$ *(possibly outside the support of* Gen*), there exists no pair* $(sk, sk') \in \{0,1\}^s \times \{0,1\}^s$ *that simultaneously satisfies* $sk \neq sk'$ *and* $\mathsf{UKCheck}(pk, sk) = \mathsf{UKCheck}(pk, sk') = 1$.

One can easily verify the following lemma. We give its proof for completeness.

Lemma 7. *Any unique-key PKE scheme is a unique-message PKE scheme.*

Proof. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a unique-key PKE scheme whose key checking algorithm is $\mathsf{UKCheck}$. Since Dec is deterministic, for all $ct \in \{0,1\}^c$, $sk = sk'$ implies $\mathsf{Dec}_{sk}(ct) = \mathsf{Dec}_{sk'}(ct)$. Therefore, for all $\lambda \in \mathbb{N}$, all $pk \in \{0,1\}^p$ and all $ct \in \{0,1\}^c$, there exists no pair $(sk, sk') \in \{0,1\}^s \times \{0,1\}^s$ that simultaneously satisfies $\mathsf{UKCheck}(pk, sk) = \mathsf{UKCheck}(pk, sk') = 1$ and $\mathsf{Dec}_{sk}(ct) \neq \mathsf{Dec}_{sk'}(ct)$. Therefore, $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a unique-message PKE scheme, for which the key checking algorithm is $\mathsf{UKCheck}$ and the plaintext-recovering algorithm is the original decryption algorithm Dec. □

F Proof of Theorem 4

In this section, we give the full proof of Theorem 4.

Proof (of Theorem 4). Let $\mathsf{UMCheck}$ and Rec be the key checking algorithm and the plaintext-recovering algorithm of Π, respectively. For each $\lambda \in \mathbb{N}$, let $\mathbb{F}_\lambda$ be the set of all functions $f : \{0,1\}^p \to \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$. We define an inefficient class of stateless adversaries $\{\mathcal{A}_{\lambda,f} = (\mathbf{Tamper}_{\lambda,f}, \mathbf{Break1}_{\lambda,f}, \mathbf{Break2}_{\lambda,f})\}_{\lambda \in \mathbb{N}, f \in \mathbb{F}_\lambda}$ as follows.

$\mathbf{Tamper}_{\lambda,f}(pk, sk)$:
1. If $\mathsf{UMCheck}(pk, sk) \neq 1$, return $\perp$.
2. Compute $(r_g, r_e, m) = f(pk)$.
3. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$ and return sk'.

Break$1_{\lambda,f}(pk)$:
1. Compute $(r_g, r_e, m) = f(pk)$.
2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
3. Return $ct = \mathsf{Enc}_{pk'}(m; r_e)$.

Break$2_{\lambda,f}(pk, m', ct^*)$:
1. Compute $(r_g, r_e, m) = f(pk)$.
2. If $m = m'$, do an exhaustive search to find the lexicographically first sk such that $\mathsf{UMCheck}(pk, sk) = 1$, and return $m^* = \mathsf{Rec}_{sk}(ct^*)$. If $m \neq m'$ or such sk does not exist, return $\bot$.

It is straightforward that for each $f \in \mathbb{F}_\lambda$, $\mathcal{A}_{\lambda,f}$ breaks the WTR security with advantage 1, due to the correctness of Π and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Rec})$, and the unique-message property of Π.

We now define the PPT stateful simulator $\mathbf{Sim}(1^\lambda)$. It internally keeps a list Ł (which is initially empty) as a state, and answers queries as follows.

- On receiving a **Tamper** query (pk, sk):
  1. If $\mathsf{UMCheck}(pk, sk) \neq 1$, return $\bot$.
  2. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If $\overline{sk} = \bot$, replace $\overline{sk}$ with sk. If the searching process failed (i.e., pk is not currently in Ł), randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add (pk, sk, r_g, r_e, m) to Ł.
  3. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$ and return sk'.
- On receiving a **Break**1 query pk:
  1. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If the searching process failed, then randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add $(pk, \bot, r_g, r_e, m)$ to Ł.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. Return $ct = \mathsf{Enc}_{pk'}(m; r_e)$.
- On receiving a **Break**2 query (pk, m', ct^*):
  1. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If the searching process failed or $\overline{sk} = \bot$, return $\bot$.
  2. If $m = m'$, return $m^* = \mathsf{Rec}_{\overline{sk}}(ct^*)$. Otherwise, return $\bot$.

Let $\mathcal{B}^{(\cdot)}$ be any oracle-access PPT machine. We show that $|\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$ by giving hybrid machines.

Hybrid machine $\mathcal{A}_0$: We define an inefficient *stateful* machine $\mathcal{A}_0$ as follows.

- On receiving a **Tamper** query (pk, sk):
  1. If $\mathsf{UMCheck}(pk, sk) \neq 1$, return $\bot$.
  2. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł (where Ł is initialized with $\emptyset$). If $\overline{sk} = \bot$, replace $\overline{sk}$ with sk. If the searching process failed, randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add (pk, sk, r_g, r_e, m) to Ł.
  3. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$ and return sk'.
- On receiving a **Break**1 query pk:
  1. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If the searching process failed, then randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add $(pk, \bot, r_g, r_e, m)$ to Ł.
  2. Compute $(pk', sk') = \mathsf{Gen}(1^\lambda; r_g)$.
  3. Return $ct = \mathsf{Enc}_{pk'}(m; r_e)$.

– On receiving a **Break**2 query (pk, m', ct^*):
  1. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If the searching process failed, then randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add $(pk, \perp, r_g, r_e, m)$ to Ł.
  2. If $m = m'$, do an exhaustive search to find the lexicographically first sk such that $\mathsf{UMCheck}(pk, sk) = 1$, and return $m^* = \mathsf{Rec}_{sk}(ct^*)$. If $m \neq m'$ or such sk does not exist, return $\perp$.

Lemma 8. $\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] = \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)]$.

Proof (of Lemma 8). Since f (used by $\mathcal{A}_{\lambda,f}$) is randomly sampled from $\mathbb{F}_\lambda$, the outputs of f are perfect randomness in the view of $\mathcal{B}$. Moreover, what $\mathcal{A}_0$ does is just simulating f by randomly sampling its outputs. Then this lemma follows from the fact that the views of $\mathcal{B}$ in the interactions with $\mathcal{A}_{\lambda,f}$ (where $f \leftarrow \mathbb{F}_\lambda$) and with $\mathcal{A}_0$ are identical. □

Hybrid machine $\mathcal{A}_1$: We now define an inefficient stateful machine $\mathcal{A}_1$ which is the same as $\mathcal{A}_0$, except that on receiving a **Break**2 query (pk, m', ct^*), it runs as follows. (Bellow, the difference from $\mathcal{A}_0$ is only in Step 2, and is *emphasized.*)
1. Search $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł. If the searching process failed, then randomly choose $(r_g, r_e, m) \leftarrow \mathcal{R}_g \times \mathcal{R}_e \times \mathcal{M}$, and add $(pk, \perp, r_g, r_e, m)$ to Ł.
2. If $m = m'$, *compute* $m^* = \mathsf{Rec}_{\overline{sk}}(ct^*)$ *(if* $\overline{sk} \neq \perp$*), or do an exhaustive search to find the lexicographically first* sk *such that* $\mathsf{UMCheck}(pk, sk) = 1$ *(if* $\overline{sk} = \perp$*), and return* $m^* = \mathsf{Rec}_{sk}(ct^*)$. If $m \neq m'$ or such sk does not exist, return $\perp$.

Lemma 9. $\Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)] = \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)]$.

Proof (of Lemma 9). This lemma follows from the fact that $\mathsf{Rec}_{sk}(ct^*) = \mathsf{Rec}_{\overline{sk}}(ct^*)$ holds for all $ct^* \in \{0,1\}^c$ and all $(sk, \overline{sk})$ such that $\mathsf{UMCheck}(pk, sk) = \mathsf{UMCheck}(pk, \overline{sk}) = 1$, due to the unique-message property of Π, and $\mathsf{UMCheck}(pk, \overline{sk}) = 1$ holds for all $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł where $\overline{sk} \neq \perp$ during the execution of $\mathcal{B}^{\mathcal{A}_0(1^\lambda)}(1^\lambda)$ and $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$, due to the check done in the first step in the response to a **Tamper** query. □

Now we briefly recall the difference between **Sim** and $\mathcal{A}_1$. Taking as input a **Break**2 query (pk, m', ct^*), **Sim** runs in the same way as $\mathcal{A}_1$ does, except that when searching $(pk, \overline{sk}, r_g, r_e, m) \in$ Ł fails or $\overline{sk} = \perp$, **Sim** returns $\perp$. Based on this observation, we show the indistinguishability between the interactions with **Sim** and with $\mathcal{A}_1$.

Lemma 10. $|\Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$.

Proof (of Lemma 10). Let E be the event that during the execution of $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$, $\mathcal{B}$ makes a **Break**2 query (pk, m', ct^*) such that
– pk has not appeared in a valid **Tamper** query (the answer of which is not $\perp$) previously, and

– the answer to this **Break**2 query is *not* $\perp$.

If E does not occur during the execution of $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$, then the view of $\mathcal{B}$ is identical to its view in the interaction with **Sim**. As a result, we have $|\Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)]| \leq \Pr[E]$. It remains to show that $\Pr[E]$ is negligible.

Claim 2 *Let the total number of (all kinds of) queries made by $\mathcal{B}$ be q. Then there exists a PPT adversary $\mathcal{E}$ that breaks the weak one-wayness of Π with probability at least $\Pr[E]/q^2$.*

Proof idea of Claim 2. A tampered key is generated as $(pk', sk') \leftarrow \mathsf{Gen}(1^\lambda; r_g)$ where r_g is a randomness linked with a public key pk. If $\mathcal{B}$ does not make a valid **Tamper** query (which does not lead $\mathcal{A}_1$ to output $\perp$) including pk, it would learn no information on (pk', sk'). In this case, making a **Break**2 query, such that pk is included in this query but $\mathcal{A}_1$ does not output $\perp$, means decrypting a randomly generated ciphertext, due to the check done at the second step in the response to a **Break**2 query. Therefore, $\mathcal{B}$ can be used to break the weak one-wayness of Π. The formal proof is as follows.

Proof (of Claim 2). The description of $\mathcal{E}$ is as follows.

The challenger samples $(pk', sk') \leftarrow \mathsf{Gen}(1^\lambda)$, chooses $m \leftarrow \mathcal{M}$, computes $ct \leftarrow \mathsf{Enc}_{pk'}(m)$, and gives $(1^\lambda, ct)$ to $\mathcal{E}$. Then $\mathcal{E}$ randomly chooses $\hat{i}, \hat{j} \leftarrow \{1, \cdots, q\}$, and interacts with $\mathcal{B}$ in the same way as $\mathcal{A}_1$ does (during the execution of $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$), except that

1. If $\mathcal{E}$ needs to do an exhaustive search, it returns $\perp$ to $\mathcal{B}$.
2. From the $\hat{i}$th query, every time when receiving a **Tamper** or **Break**2 query (respectively, a **Break**1 query) including pk used in the $\hat{i}$th query, $\mathcal{E}$ returns $\perp$ (respectively, ct) to $\mathcal{B}$ (except for the $\hat{j}$th query).[22]
3. On receiving the $\hat{j}$th query, if it is a **Break**2 query denoted by (pk, m', ct^*), $\mathcal{E}$ returns m' to the challenger and terminates. Otherwise, $\mathcal{E}$ aborts.

Now we argue that $\mathcal{E}$ returns m' such that $m = m'$ with probability at least $\Pr[E]/q^2$.

During the execution of $\mathcal{B}^{\mathcal{A}_1(1^\lambda)}(1^\lambda)$, when E occurs, $\mathcal{B}$ must have made a **Break**2 query (pk, m', ct^*), such that (a) there exists no entry $(pk, \overline{sk}, r_g, r_e, m) \in \mathtt{L}$ such that $\overline{sk} \neq \perp$, but (b) $\mathcal{A}_1$ does not return $\perp$ as its answer. Therefore, for randomly sampled $\hat{i}, \hat{j} \leftarrow \{1, \cdots, q\}$ (not learnt by $\mathcal{B}$), when E occurs, the probability that

– the $\hat{j}$th query is the first **Break**2 query satisfying the above two conditions (a) and (b), and
– the $\hat{j}$th query includes pk, which is firstly used in the $\hat{i}$th query,

is at least $1/q^2$. Since $\mathcal{E}$ perfectly simulates $\mathcal{A}_1$ in the interaction with $\mathcal{B}$ in this case (till the termination), it returns m' such that $m' = m$ with probability at least $\Pr[E]/q^2$. Notice that in this case, r_g, r_e, and m linked with pk (used in the $\hat{j}$th query) are not used by $\mathcal{E}$, and we view (pk', sk') and ct (where (pk', sk')

[22] $\mathcal{E}$ may terminate before receiving the $\hat{i}$th query.

and ct are generated by the challenger and (pk', sk') is not learnt by $\mathcal{E}$) as the key pair and ciphertext generated by using (r_g, r_e, m). This completes the proof of Claim 2. □

Due to the assumption that Π is weakly one-way, $\Pr[E]/q^2$ is negligible. Therefore, $\Pr[E]$ is negligible, completing the proof of Lemma 10. □

Due to Lemmas 8 to 10, we have $|\Pr_{f \leftarrow \mathbb{F}_\lambda}[1 \leftarrow \mathcal{B}^{\mathcal{A}_{\lambda,f}}(1^\lambda)] - \Pr[1 \leftarrow \mathcal{B}^{\mathbf{Sim}(1^\lambda)}(1^\lambda)]| \leq negl(\lambda)$, completing the proof of Theorem 4. □

Table of Contents