From Random Oracles to Ideal Signatures, and Back

Cong Zhang
University of Maryland
czhang20@umd.edu

Hong-Sheng Zhou
Virginia Commonwealth University
hszhou@vcu.edu

May 3, 2021

Abstract

We investigate the digital signature schemes in the indifferentiability framework. We show that the well-known Lamport one-time signature scheme, and the tree-based signature scheme can be "lifted" to realize ideal one-time signature, and ideal signature, respectively, without using computational assumptions. We for first time show that the ideal signatures, ideal one-time signatures, and random oracles are equivalent in the framework of indifferentiability.

Contents

1 Introduction

The notion of indifferentiability was introduced by Maurer et al. [MRH04], as a generalization of indistinguishability tailored to settings where internal state is publicly available. The indifferentiability framework comes with a composition theorem, supporting modular security analysis for cryptographic constructions. It has been proven that the indifferentiability framework is very useful in practice. To see this, let's consider the provably secure cryptographic schemes in the random oracle (RO) model [BR93]. Many practical cryptographic schemes have achieved provable security in the RO model. However, due to length-extension attacks, many hash function constructions cannot be modeled as a RO, even the underlying building blocks can be modeled as an ideal primitive such as an ideal compression function. In these attacks, the adversaries explore the weakness of the structure of the iterative hash functions. We note that, even we can construct hash functions against known length-extension attacks, it remains unclear if the resulting functions can defend against other unknown attacks.

Using the indifferentiability framework, Coron et al. [CDMP05] propose a composable approach to design hash functions that "behave like" random oracles. More concretely, consider a hash function construction $H^{\mathbf{h}}$ where $\mathbf{h}$ is an (ideal) primitive. We now just need to show that the construction $H^{\mathbf{h}}$ can be indifferentiable from a RO. After that, several other variants were proposed and studied in [DP06, DP07]. In addition, a line of notable work applied the framework to the ideal cipher, and showed that the Feistel construction is indifferentiable from an ideal cipher, see [CHK$^+$16, DKT16, DS16]. Authenticated encryption has also been rigorously investigated in [BF18].

Very recently, Zhandry and Zhang [ZZ20] have made efforts, for the first time, to design *indifferentiable pubic key cryptographic primitives*. They demonstrated several interesting feasibility results. They define the idealized model for public key encryption (PKE), i.e., ideal PKE, in the indifferentiability framework, and present a construction which is indifferentiable from ideal PKE, in the random oracle model, *by using computational assumptions*. Note that, computational assumptions indeed are necessary, facing the theoretical barrier of Impagliazzo and Rudich [IR89].

In the same paper [ZZ20], Zhandry and Zhang further study digital signatures. They define the idealized model for digital signatures, i.e., ideal signatures, but with additional *uniqueness* property specified, and then consider the constructions. Our first observation is that their definition for ideal signatures in [ZZ20] is *not* complete: verifications for certain tuples of (public-key, message, signature) are *not* defined. As a consequence, a differentiator can exploit this weakness to tell the difference between the real and the ideal worlds, and thus their corresponding proof of security fails. (More elaborations can be found on next page, Section 1.2.1.)

This motivates us to study the (ordinary) digital signatures in the indifferentiability framework, from definitions to constructions. We want to make it explicit that, our focus here is not the unique signatures as in [ZZ20]. Instead, we are seeking to understand (and hopefully then be able to provide an affirmative answer to) the following fundamental question for ordinary digital signatures:

> *Is that possible to show the equivalence of the ideal signatures and the random oracle in the indifferentiability framework?*

Note that, different from PKE or unique signatures, it is known that, ordinary digital signatures is in Minicrypt [Rom90], following the conventional property based definitions. We are interested in investigating whether the complexity landscape of digital signatures remains the same or not in the indifferentiability framework.

1.1 Our results

We give an affirmative answer to the above question. More concretely, we have the following results.

Analyzing the definitions and constructions in [ZZ20]: We start with a security analysis of the signature by Zhandry and Zhang [ZZ20], as presented in Section 3.1. More concretely, we point out that the definition of ideal signature in [ZZ20] is incomplete, which allows a trivial attack; while this definition issue can be fixed, we further illustrate that the signature design can*not* be proven to achieve indifferentiability: a differentiator breaks the indifferentiability with overwhelming advantage.

Defining ideal (one-time) signatures: We provide definitions for ideal signature and for ideal one-time signature, respectively, in the indifferentiability framework. In our definitions, only natural queries and responses are enabled by the new ideal primitives. We remark that, this is the first effort for defining ideal (ordinary) signatures and ideal one-time signatures in the indifferentiability framework.

Indifferentiable one-time signatures from random oracles: We provide a construction for indifferentiable one time signature in the random oracle model (without using any additional assumption). More concretely, we demonstrate that, a simple and natural variant of the well-known Lamport signatures can be proven indifferentiable from the ideal one-time signature that we defined.

Indifferentiable signatures from indifferentiable one-time signatures: We provide a construction for indifferentiable signature, using indifferentiable one time signature (without any additional assumption). Our construction is based on the tree-based construction [Gol04], which uses two building blocks, a one time signature scheme and pseudorandom function (PRF). While the building blocks can be replaced with suitable ideal primitives, i.e., ideal one-time signatures and random oracles, the resulting scheme is a deterministic signature scheme. We will explain in next subsection the difficulty of realizing ideal signatures by using deterministic signature scheme. Fortunately, we develop a beautiful strategy, called "partially randomized signing", to resolve the difficulty.

Completing the picture. Finally, we provide the additional contribution of constructing indifferentiable random oracle model from ideal signatures or ideal one-time signatures (without any additional computational assumptions). An immediate corollary of this additional result is that, ideal signatures, ideal one-time signatures, and random oracles are equivalent in indifferentiability framework. This can be viewed as a *composable analog* of the equivalence of signatures and one-way functions [Rom90].

1.2 Our techniques

To achieve the above listed results, we have to resolve multiple technical difficulties. In this subsection, we describe our techniques.

1.2.1 Analyzing the definition and construction in [ZZ20].

In [ZZ20], Zhandry and Zhang provide a definition for ideal signatures; details can also be found in Section 3.1. in their definition [ZZ20], first, consider signing-key space $\mathcal{SK}$, verification-key space $\mathcal{PK}$, message space $\mathcal{M}$, and signature space $\mathbf{\Sigma}$; note that, key generation, signing, and verification, can be viewed as injections which can be defined over spaces $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{M}$, $\mathbf{\Sigma}$, properly; let $\mathbf{G}$,$\mathbf{S}$, and $\mathbf{V}$ be the sets of injections for key generation, signing, and the set of predicates for verification, respectively. Based on these spaces and injection sets, then a set $\mathbf{T}$ of function tuples $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ is defined to capture the functionality of key generation, signing, and verification; that is the following conditions must be satisfied

1. $\mathbf{Gen} \in \mathbf{G}$, $\mathbf{Sign} \in \mathbf{S}$ and $\mathbf{Verify} \in \mathbf{V}$;
2. $\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}, \mathbf{Verify}(\mathbf{Gen}(SK), M, \mathbf{Sign}(SK, M)) = 1$;
3. $\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}, V \in \mathbf{\Sigma}$, if $V \neq \mathbf{Sign}(SK, M)$, then $\mathbf{Verify}(\mathbf{Gen}(SK), M, \mathbf{Sign}(SK, M)) = 0$;
4. $\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}, V_1, V_2 \in \mathbf{\Sigma}$, if $\mathbf{Verify}(\mathbf{Gen}(SK), M, V_1) = \mathbf{Verify}(\mathbf{Gen}(SK), M, V_2) = 1$, then $V_1 = V_2$.

Now, a tuple $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, if uniformly sampled from the set $\mathbf{T}$, is defined as an ideal digital signature.

Unfortunately, as we already mentioned at the very beginning of the paper, this definition is incomplete. Specifically, let $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ be an ideal digital signature, associated with $\mathcal{SK}, \mathcal{PK}, \mathcal{M}, \mathbf{\Sigma}$. We define that *a public key $PK \in \mathcal{PK}$ is honest*, if there exists a secret key $SK \in \mathcal{SK}$ such that $\mathbf{Gen}(SK) = PK$; otherwise we say PK is dishonest. We can immediately observe that, the verification algorithm $\mathbf{Verify}(PK, \cdot, \cdot)$ is not defined, when PK is dishonest, and thus this definition is not complete. Note that, a differentiator can easily exploit this weakness in the definition to distinguish the real world from the ideal world, and fails their security proof.

One would argue that this "incompleteness" in their definition, can be naturally fixed by revising condition 3 above into the following condition 3':

3'. $\forall PK \in \mathcal{PK}, M \in \mathcal{M}, V \in \mathbf{\Sigma}$, if $\nexists SK \in \mathcal{SK}$ s.t. $PK = \mathbf{Gen}(SK)$ and $V = \mathbf{Sign}(SK, M)$, then $\mathbf{Verify}(PK, M, V) = 0$.

Note that, in this updated version of definition, if PK is dishonest, then for any message M and signature value V, we have that $\mathbf{Verify}(PK, M, V) = 0$. We justify that this definition is natural, as in signature, we indeed wish that only the honest public key, along with the corresponding message and signature, could pass the verification test.

However, the current construction in [ZZ20] cannot realize this definition. In the following, we illustrate a distinguishing attack that breaks their security proof for realizing ideal signature in [ZZ20]. Recalling their construction $\Pi = (\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify})$:

- $\Pi.\textsc{Gen}(SK) = \mathcal{P}(\Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}(\mathcal{H}_{\mathrm{sk}}(SK)))$;
- $\Pi.\textsc{Sign}(SK, M) = \mathcal{E}(\Pi.\textsc{Gen}(SK)||M, \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}(\mathcal{H}_{\mathrm{sk}}(SK), \mathcal{H}_{\mathrm{msg}}(M)))$;
- $\Pi.\textsc{Verify}(PK, M, V) = \Pi_{\mathsf{sm}}.\textsc{Verify}_{\mathsf{sm}}(\mathcal{P}^{-1}(PK), \mathcal{H}_{\mathrm{msg}}(M), \mathcal{E}^{-1}(PK||M, V))$,

where $\mathcal{H}_{\mathrm{sk}}$ and $\mathcal{H}_{\mathrm{msg}}$ are two random oracle models, $\mathcal{P}$ is a random permutation, $\mathcal{E}$ is an ideal cipher model. $\Pi_{\mathsf{sm}} = (\Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}, \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}, \Pi_{\mathsf{sm}}.\textsc{Verify}_{\mathsf{sm}})$ is a standard-model signature scheme satisfies: 1) uniqueness; 2) pseudorandom public keys; 3) random-message-attack security. To describe our distinguishing attack, we first build an alternative signature $\Pi_{\mathsf{sm\text{-}magic}} = (\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Gen}_{\mathsf{sm\text{-}magic}}, \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Sign}_{\mathsf{sm\text{-}magic}}, \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm\text{-}magic}})$ in the standard model. Concretely, let $\mathcal{PK}_{\mathsf{sm}}/\mathcal{SK}_{\mathsf{sm}}$ be the public/secret key space of Π_{sm}, and $pk_{\mathsf{sm\text{-}magic}}$ is a magic public key such that $pk_{\mathsf{sm\text{-}magic}} \notin \mathcal{PK}_{\mathsf{sm}}$, then we define the public/secret key space ($\mathcal{PK}_{\mathsf{sm\text{-}magic}}/\mathcal{SK}_{\mathsf{sm\text{-}magic}}$) of $\Pi_{\mathsf{sm\text{-}magic}}$ as

$$\mathcal{PK}_{\mathsf{sm\text{-}magic}} := \mathcal{PK}_{\mathsf{sm}} \cup \{pk_{\mathsf{sm\text{-}magic}}\}; \mathcal{SK}_{\mathsf{sm\text{-}magic}} := \mathcal{SK}_{\mathsf{sm}}.$$

Then we define $\Pi_{\mathsf{sm\text{-}magic}}$ as:

- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Gen}_{\mathsf{sm\text{-}magic}}\ (SK) = \Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}\ (SK)$;
- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Sign}_{\mathsf{sm\text{-}magic}}\ (SK, M) = \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}\ (SK, M)$;
- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm\text{-}magic}}\ (PK, M, V) = \begin{cases} \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm}}(PK, M, V) & \text{if } PK \neq pk_{\mathsf{sm\text{-}magic}}, \\ 1 & \text{if } PK = pk_{\mathsf{sm\text{-}magic}}. \end{cases}$

Trivial to note that $\Pi_{\mathsf{sm\text{-}magic}}$ also satisfies uniqueness, pseudorandom public keys, and random-message attack security, thus according to [ZZ20], we have that the signature construction, $\Pi_{\mathsf{magic}} = (\Pi_{\mathsf{magic}}.\textsc{Gen}, \Pi_{\mathsf{magic}}.\textsc{Sign}, \Pi_{\mathsf{magic}}.\textsc{Verify})$, that uses $\Pi_{\mathsf{sm\text{-}magic}}$ as a building block is also indifferentiable from an ideal signature. Next, in Figure 1, we describe a differentiator $\mathcal{D}$ to break the security of Π_{magic}:

Differentiator in real world $\mathcal{D}_{\mathsf{real}}$	Differentiator in ideal world $\mathcal{D}_{\mathsf{ideal}}$
$M \twoheadleftarrow \mathcal{M}$; $V \twoheadleftarrow \mathbf{\Sigma}$;	$M \twoheadleftarrow \mathcal{M}$; $V \twoheadleftarrow \mathbf{\Sigma}$;
$PK \leftarrow \mathcal{P}(pk_{\mathsf{sm\text{-}magic}})$;	$PK \leftarrow \mathcal{P}(pk_{\mathsf{sm\text{-}magic}})$;
return $\Pi_{\mathsf{magic}}.\textsc{Verify}_{\mathsf{sm\text{-}magic}}(PK, M, V)$.	return $\mathbf{Verify}(PK, M, V)$.

Figure 1: Differentiator for Π_{magic}.

Note that, in the real world, the differentiator always outputs 1, due to the definition of $pk_{\mathsf{sm\text{-}magic}}$. However in the ideal world, if the simulator responds to the query with a honest public key (same as the proof in [ZZ20]), then the differentiator would output 0 with overwhelming probability, as the message M and signature value V are randomly sampled; and if the simulator responds to the query with a *dishonest* public key, then by definition, we have that differentiator would always output 0.

1.2.2 Defining ideal (one-time) signatures.

We provide the first definitions for ideal signature and for ideal one-time signature in the indifferentability framework. Previous formulation for ideal signature by Zhandry and Zhang [ZZ20] (also see Section 3.1.1), captures the uniqueness property. However, our goal here is to capture the idealized version of ordinary signatures; note that to achieve the additional property of uniqueness, we need to pay the price of computational assumptions.

We follow the presentation style of Ristenpart et al [RSS11]; in our formulation of the ideal signature (in Figure 5), we define three sub-procedures, $\mathbf{Gen}.hon$, $\mathbf{Sign}.hon$, $\mathbf{Verify}.hon$, together to form the "honest" interface, and the "adversarial" interface is defined to be identical to the honest interface. In the ideal signature, the response values for PK and for signature V are randomly sampled. Whenever a signature is generated, the involved signing-key SK must be well-defined. In addition, through both honest and adversarial interfaces, the same signing-key SK is allowed to be used for signing multiple distinct messages.

To define ideal *one-time* signature (in Figure 6), we introduce an idea of defining the honest and the adversarial interfaces differently. While the adversarial is identical to the adversarial/honest interface of the ideal signature, capturing the intuition that the same signing-key SK is allowed to be used for signing multiple distinct messages. In contrast, in the honest interface of ideal one-time signature, the same SK for signing different messages, must be disallowed. We achieve this through a careful bookkeeping strategy.

1.2.3 Constructing indifferentiable one-time signatures.

Our first construction is a one-time signature scheme which is indifferentiable from the ideal one-time signature. The starting point of the construction is Lamport's one-time signature scheme. In Lamport's design[1], a one-way function f is used in the key generation algorithm; more concretely, the signing key $\boldsymbol{sk}$ consists of an n-by-2 matrix of random strings denoted as, $\boldsymbol{sk} = \langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle$, while the corresponding verification key $\boldsymbol{pk}$ also consists of an n-by-2 matrix of strings, i.e., $\boldsymbol{pk} = \langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle$, where $pk_{i,0} = f(sk_{i,0})$ and $pk_{i,1} = f(sk_{i,1})$ for all $1 \leq i \leq n$. A collision resistant hash function H is used in the signing algorithm; to sign a given message m, first the message is compressed into a n-bit string $b_1 b_2 \cdots b_n \leftarrow H(\boldsymbol{pk}, m)$, where $b_i \in \{0,1\}$; the corresponding signature $\boldsymbol{\sigma}$ consists of "half of the signing key", i.e., $\boldsymbol{\sigma} = sk_{1,b_1}||\cdots||sk_{n,b_n}$. Finally, in the verification algorithm, upon receiving a message m and signature $\boldsymbol{\sigma} = \sigma_1||\cdots||\sigma_n$, first compute $b_1 b_2 \cdots b_n \leftarrow H(\boldsymbol{pk}, m)$; if it holds that, $f(\sigma_i) = pk_{i,b_i}$, for all $i = 1, \ldots, n$, then the message-signature pair $(m, \boldsymbol{\sigma})$ is valid.

Construction ideas. One straightforward way to build an indifferentiable one-time signature is to apply the techniques in [ZZ20], however, due to the dishonest public key attack (subsection 1.2.1), it seems unclear that such a construction is indifferentiably-secure or not. On the other side, we observe that, the attack is mainly caused by the dishonest public keys, and if we can "remove" those dishonest public keys in a subtle way, then the ideas in [ZZ20] might be sufficient.

Eliminating dishonest public key. We observe that, if every public key value in the public key space is honest with overwhelming probability, then the dishonest public keys are "removed". Following this observation, we have that, with high probability, for any public key PK, there exists at least one secret key SK such that $\mathbf{oGen}(SK) = PK$. Therefore, the secret key space $\mathcal{SK}$ would be larger than the public key space $\mathcal{PK}$, and thus any function that maps $\mathcal{SK}$ to $\mathcal{PK}$ $\mathbf{Gen}$ cannot be an injection. In this work, we update the definition by setting $|\mathcal{SK}| \gg |\mathcal{PK}|$, and $\mathbf{oGen}(\cdot)$ to a random function (rather than an injection comparing to the definition in [ZZ20]). Next, we show how to combine this idea with the techniques in [ZZ20].

Recalling the indifferentiable digital signature schemes $\Pi = (\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify})$ in [ZZ20]:

- $\Pi.\textsc{Gen}(SK) = \mathcal{P}(\Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}(\mathcal{H}_{\text{sk}}(SK)))$;
- $\Pi.\textsc{Sign}(SK, M) = \mathcal{E}(\Pi.\textsc{Gen}(SK)||M, \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}(\mathcal{H}_{\text{sk}}(SK), \mathcal{H}_{\text{msg}}(M)))$;
- $\Pi.\textsc{Verify}(PK, M, V) = \Pi_{\mathsf{sm}}.\textsc{Verify}_{\mathsf{sm}}(\mathcal{P}^{-1}(PK), \mathcal{H}_{\text{msg}}(M), \mathcal{E}^{-1}(PK||M, V))$,

Known that Lamport's one-time signature is built on one way function and collision resistant hash function, to get rid of the computational assumptions we first upgrade the Lamport's scheme by replacing the one way function f and collision resistant hash function H with two random oracle $\mathcal{H}_{\text{OneWay}}$ and $\mathcal{H}_{\text{position}}$, respectively. Now the above $pk_{i,0} = f(sk_{i,0})$ and $pk_{i,1} = f(sk_{i,1})$ are upgraded into $pk_{i,0} = \mathcal{H}_{\text{OneWay}}(sk_{i,0})$ and $pk_{i,1} = \mathcal{H}_{\text{OneWay}}(sk_{i,1})$, and the position $b_1 b_2 \cdots b_n \leftarrow H(\boldsymbol{pk}, m)$ is upgraded into $b_1 b_2 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(\boldsymbol{pk}, m)$.

[1]Please also see Goldreich's [Gol04] and Katz-Lindell's textbook [KL07] for a clear illustration.

Next, we give the high level intuition to eliminate the dishonest public key. Specifically, we set that $|SK| = 8n\lambda$, $|sk_{i,b}| = 2\lambda$, $|pk_{i,b}| = \lambda$ and for the random oracles $\mathcal{H}_{\text{sk}}$ and $\mathcal{H}_{\text{OneWay}}$, we set that $\mathcal{H}_{\text{sk}} : \{0,1\}^{8n\lambda} \to \{0,1\}^{4n\lambda}$ and $\mathcal{H}_{\text{OneWay}} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. Note that, both oracles shrinks the inputs and thus, in either real world or ideal world, there is no dishonest public key with overwhelming probability.

Next, following the strategy of [ZZ20], we roughly explain the intuition why we also need $\mathcal{P}$ and $\mathcal{E}$ in our construction.

Why random permutation ($\mathcal{P}$, $\mathcal{P}^{-1}$)? As described above, in our design, $PK = \mathcal{P}(\boldsymbol{pk})$. Using random permutation to eliminate the "matrix" structure of the $\boldsymbol{pk}$ in Lamport's design, is essential for our construction. One main intuition behind the ideal one-time signature is that, *when a valid signature is generated, the signer must be aware of a well-defined signing key.* However, if we do not use random permutation to mask the $\boldsymbol{pk}$, say now $PK = \boldsymbol{pk}$, it is possible to construct a signature which can pass the verification, without having the signing key clearly defined! More concretely, the adversary/differentiator can develop the following correlation attack: First, the differentiator based on a well-defined signing key SK, obtains $\boldsymbol{sk} = \langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle$ and $\boldsymbol{pk} = \langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle$. Note that $\boldsymbol{pk} = PK = \textbf{oGen}(SK)$. Now the differentiator develops a $\boldsymbol{pk}'$ which is correlated to $\boldsymbol{pk}$, say $PK' = \boldsymbol{pk}' = \langle pk^*_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle$ where $pk^*_{1,0}$ is randomly sampled, *without even being aware of the corresponding* $sk^*_{1,0}$. The differentiator then chooses a message M so that $\mathcal{H}_{\text{position}}(PK', M) = 1b_2 \cdots b_n$, i.e., $b_1 = 1$. The signature for such M is $\sigma = \mathcal{E}\Big(PK'||M,\ sk_{1,1}||sk_{2,b_2}|| \cdots ||sk_{n,b_n}\Big)$. Note that, the signature σ can pass the verification; however, the underlying signing key for such σ is known, which deviates from the definition of ideal one-time signature.

Why ideal cipher ($\mathcal{E}$, $\mathcal{E}^{-1}$)? The ideal cipher is also essential in our design. In Lamport's original design, the signature consists of half of the $\boldsymbol{sk}$; denoted as $\langle sk_{1,b_1}|| \cdots ||sk_{n,b_n}\rangle$. In our construction, we first pad dummy string $0 \cdots 0$ to the signature to make the length of signature to be sufficiently long, then apply the ideal cipher model to "encapsulate" the internal signature $(sk_{1,b_1}||sk_{2,b_2}|| \cdots ||sk_{n,b_n})$. If the ideal cipher is not used, say we let $\sigma = \langle sk_{1,b_1}||sk_{2,b_2}|| \cdots ||sk_{n,b_n}\rangle$, then the differentiator can launch the following attack. First, the differentiator samples SK and M, and obtains PK and σ via the honest interfaces **oGen** and **oSign**, respectively. Concretely, $PK = \textbf{oGen}(SK)$ and $\sigma = \textbf{oSign}(SK, M) = \langle sk_{1,b_1}||sk_{2,b_2}|| \cdots ||sk_{n,b_n}\rangle$. After that, the differentiator makes queries $\mathcal{P}^{-1}(PK)$ and $\mathcal{H}_{\text{position}}(PK, M)$, and thus obtain the corresponding pk and $b_1 \ldots b_n$. Note that, in the real world, it's apparent that $\mathcal{H}_{\text{OneWay}}(sk_{i,b_i}) = pk_{i,b_i}$, meanwhile, the simulator knows nothing of sk (it only knows PK). Then, the differentiator would proceed the following test: it flips a coin b, and samples $\widehat{sk}$ such that $|\widehat{sk}| = |sk_{1,b_1}|$, and make a query $\mathcal{H}_{\text{OneWay}}(\widehat{sk})$ if $b = 0$ and $\mathcal{H}_{\text{OneWay}}(sk_{1,b_1})$ if $b = 1$. After receiving the response $\widehat{pk}$, the differentiator:

- if $b = 0$, returns $(\widehat{pk} \overset{?}{\neq} pk_{1,b_1})$[2],
- if $b = 1$, returns $(\widehat{pk} \overset{?}{=} pk_{1,b_1})$.

Easy to note that in the real world, the differentiator returns 1 with high probability (assuming $|pk_{1,b_i}|$ is sufficiently long). However, in the ideal world, the simulator would fail with at least probability $\frac{1}{2}$. In fact, although the simulator knows the value pk_{1,b_1}, it has no knowledge of sk_{1,b_1} which means it cannot differ sk_{1,b_1} and $\widehat{sk}$ and thus cannot guess the coin b. As a result, the simulator would fail to respond to the query $\mathcal{H}_{\text{OneWay}}(\widehat{sk})$ or $\mathcal{H}_{\text{OneWay}}(sk_{1,b_1})$ with at least probability $\frac{1}{2}$.

How the ideal cipher $(\mathcal{E}, \mathcal{E}^{-1})$ helps? Encapsulating with $\mathcal{E}$, the signature then will be with form of $\sigma = \mathcal{E}(PK||M, sk_{1,b_1}||sk_{2,b_2}|| \cdots ||sk_{n,b_n}||0 \cdots 0)$. We immediately note that given σ, the differentiator has no information of sk_{i,b_i}, and thus it can not proceed the attack above. Moreover, after applying the ideal cipher, there are only two ways for the differentiator to obtain the value sk_{i,b_i}: 1) makes a query $\mathcal{H}_{\text{sk}}(SK)$; 2) makes a query

[2]$(\widehat{pk} \overset{?}{\neq} pk_{1,b_1}) = 1$ iff $\widehat{pk} \neq pk_{1,b_1}$.

$\mathcal{E}^{-1}(PK||M,\sigma)$. In the first case, the simulator would know both SK and sk; in the second one, the simulator would randomly sample sk itself, and implicitly set $\mathcal{H}_{\mathrm{sk}}(SK) = sk$. In other words, applying $(\mathcal{E},\mathcal{E}^{-1})$ will enforce to differentiator to give more power to the simulator so that it can complete the simulation properly.

1.2.4 Constructing indifferentiable signatures.

The starting point here is the tree-based signature scheme[3], which lift one-time security to full security. Specifically, the tree-based signature scheme (Gen, Sign, Verify), uses a pseudorandom function $F_r(\cdot)$ and a one-time signature scheme (oGen, oSign, oVerify) as building blocks. We consider a full binary tree of depth n in the construction; and the basic idea in the tree-based construction is to use the verification- and signing-key (of a one-time signature scheme) to authenticate/sign two fresh instances (of the one-time signature scheme), and then use each of the instances to authenticate/sign two fresh instances, and so on. In this way, a binary tree of fresh instances of the one-time signature will be formed, in which each internal node authenticates its two children. The leaves of this tree will be used to sign actual messages, and here each leaf is used at most once with high probability. Concretely, to sign a message, the resulting signature consists of (1) a one-time signature to the message which is authenticated with respect to the verification-key of a leaf, and (2) an authenticated path from the root to this leaf, i.e., a sequence of one-time verification-keys of all nodes in the path, in which each such verification-key is authenticated with respect to the verification-key of its parent. To get rid of the computational assumptions, we first upgrade the tree based signature: the underlying one-time signature (oGen,oSign,oVerify) will be replaced by ideal one-time signature $\mathbf{OSIG} = (\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$; the PRF $F_r(\cdot)$ will be replaced by a random oracle $\mathcal{H}(r||\cdot)$.

Same as the case in constructing indifferentiable one-time signatures, the trivial attempt is to apply the construction in [ZZ20] by setting Π_{sm} to be the tree-based signature scheme, concretely, the construction of the indifferentiable signature should be:

$$\textsc{Gen}(SK) = \mathbf{oGen}(\text{“tree-based signing key”})$$
$$\textsc{Sign}(SK,M) = \mathcal{E}(PK||M,\ \text{“tree-based signature”}).$$

Unfortunately, there are at least two barriers: 1) the dishonest public key attack; 2) the tree based scheme is not a unique signature and the construction in [ZZ20] only works when Π_{sm} is unique signature. For the former one, if we apply the same technique as above-eliminating dishonest public key, by setting SK to be sufficiently long, then the barrier should be overcome. However the situation for the latter one is much involved, and in the following we will elaborate it carefully.

Nonce-abuse attack. Although we upgrade the tree-based scheme to be information-theoretically secure and eliminate the dishonest public key, the current construction fails to achieve indifferentiability. Intuitively, we note that the tree-based scheme is deterministic if the adversary follows the algorithm and uses the nonce by querying $\mathcal{H}(r||\cdot)$ (we call such a nonce to be honest nonce, and others to be dishonest nonce); see footnote 3, meanwhile, the verification phase would not detect whether the signature is generated by an honest nonce or a dishonest one. Thus, in the tree-based scheme, once having the signing key, the adversary can output two distinct valid signatures for the same message, by abusing different nonces.

Next, we illustrate a differentiator $\mathcal{D}$ that abuses the nonce as follows: first $\mathcal{D}$ samples SK, M and makes a query $sk \leftarrow \mathcal{H}_{\mathrm{sk}}(SK)$ (we denote sk to be the tree-based signing key for ease); then $\mathcal{D}$ computes two valid signature for M by using (sk, R) and (sk, R'), denoted as v and v' (with high probability $v \neq v'$). After that, the differentiator makes two queries $V \leftarrow \mathcal{E}(PK||M,v)$ and $V' \leftarrow \mathcal{E}(PK||M,v')$ and outputs 1 if and only if 1) $\textsc{Verify}(PK,M,V) = 1$; 2) $\textsc{Verify}(PK,M,V') = 1$; 3) $V \neq V'$.

Trivial to note that, in the real world, $\mathcal{D}$ outputs 1 as long as $v \neq v'$. In fact, $\mathcal{E}$ is an ideal cipher and for any $v \neq v'$, it's apparent that

$$\mathcal{E}(PK||M,v) \neq \mathcal{E}(PK||M,v').$$

However, in the ideal world, we observe that both v and v' are valid signatures, thus according to the definition of indifferentiability, V and V' must both pass the verification procedure. But *the simulator only knows exactly one*

[3]More concretely, the starting point is Construction 6.4.16 in Goldreich's textbook [Gol04], and the underlying one-time signature scheme is deterministic.

value that passes the verification, which is $\textsc{Sign}(SK, M)$. Thus, the simulator cannot output two distinct $V \neq V'$ that pass the verification, which refers to $\mathcal{D}$ will outputs 0 with high probability.

Partially randomized signing. To prevent this attack, we have to make the signing algorithm to be randomized, thus the simulator can respond to $V \leftarrow \textsc{Sign}(SK, M, R)$ and $V \leftarrow \textsc{Sign}(SK, M, R')$, using different randomness. However, using randomized tree-based signature might be dangerous, as we are taking one-time signature as a basic primitive, and the scheme would fail immediately if it calls the signing oracle $\mathbf{oGen}(SK, \cdot)$ twice for two distinct messages.

To resolve this obstacle, we propose a novel strategy, which we called "partially randomized signing". In the high level, we add an additional layer to the tree and now the tree has $n+1$ layers. We then treat the first n layers still to be deterministic (same as above) and randomize the $(n+1)$-th layer. More specifically, given (SK, M, R), the pair (sk, pk) that's assigned to the node (belonging to the first n layer) is only determined by SK, regardless of M and R. And the pair (sk, pk) that's assigned to the leaf (belonging to the $n+1$ layer) is determined by SK, M and R. Moreover, in the signing algorithm, the authenticated path should be also determined by both M and R.

Why does this strategy help? Let $(M, R) \neq (M', R')$[4], we need to prove that for any such pairs and any fixed SK, the signing algorithm would only call the signing oracle $\textsc{Sign}(SK, \cdot)$ once. We denote $path$ and $path'$ to be the authenticated paths for (M, R) and (M', R'), respectively. Here, we denote $\textsc{ComPrefix}(path, path')$ to be the common prefix of $path$ and $path'$, for instance, if $path = 01010$ and $path' = 01001$, then $\textsc{ComPrefix}(path, path') = 010$. It's apparent that, for any $(M, R) \neq (M', R')$, if

$$\textsc{ComPrefix}(path, path') < n,$$

then the scheme would only call the signing oracle once, for fixed SK. Assuming n to be large enough, this event holds with overwhleming probability.

However, adding randomness R into the signing algorithm would make the indifferentiability much hard, as we have to mask it properly. Concretely, we denote $seed = \mathcal{H}_{\text{seed}}(SK, M, R)$ where $\mathcal{H}_{\text{seed}}$ is an additional random oracle, and we treat $seed$ as the masked nonce in the tree-based signature. Combing all together, we have our final solution:

$$\textsc{Sign}(SK, M, R) = \mathcal{E}(PK||M, \text{"partially randomized signature"}||seed).$$

1.2.5 Constructing random oracles from ideal signatures.

We now give the idea that how to construct an indifferentiable random oracle model by ideal signatures (ideal one-time signatures), and thus complete the picture. Specifically, given an ideal signature $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, we set

$$\mathcal{H}(x) = \mathbf{Gen}(x)^{5}.$$

Immediately observe that the distribution of $\mathcal{H}(x)$ is identical to uniform and to turn this into an indifferentiability proof, we show how to simulate $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, given a true random oracle model $\mathrm{RO}(\cdot)$. This, in fact, is quite straightforward. Concretely, for the query $\mathbf{Gen}(SK)$, the simulator can just respond to it with $\mathrm{RO}(SK)$. For the sign query $\mathbf{Sign}(SK, M, R)$, the simulator would just uniformly sample a string σ from the signature space $\mathbf{\Sigma}$, respond to the query with σ and record the tuple $(\mathcal{H}(SK), SK, M, R, \sigma)$ into its internal table $\mathbb{T}$. And for $\mathbf{Verify}(PK, M, \sigma)$, the simulator would go over its internal table $\mathbb{T}$ and return 1 if and only if there exists a tuple $(PK, SK, M, R, \sigma) \in \mathbb{T}$. Easy to note that, as long as the signature space $\mathbf{\Sigma}$ is sufficiently large, this simulator works properly.

Organization In the rest of our paper, Section 2 gives notations and definitions of the indifferentiability framework, in Section 3 we first analyze the signature definition and construction in [ZZ20] and point out the issues, and then present our new definitions for Ideal Signature and for Ideal one-time Signature. In Section 4, we build an indifferentiable one-time signature from random oracle and present the proof; additional proof details can be found in Appendix A. The construction of indifferentiable signature is given in Section 5.

[4] M might be equal to M'.

[5] The indifferentiable random oracle model from ideal one-time signature can be built in the same way, say $\mathcal{H}(x) = \mathbf{oGen}(x)$.

2 Preliminaries

Notation. Throughout this paper, $\lambda \in \mathbb{N}$ denotes the security parameter. For a non-empty finite set $\mathcal{X}$, we denote a uniformly random sample x from $\mathcal{X}$ as $x \twoheadleftarrow \mathcal{X}$. We overload this notation to extend to probabilistic algorithms, so that $y \twoheadleftarrow \mathcal{A}(x)$ means that y is assigned a value according to the distribution induced by algorithm $\mathcal{A}$ whose input value is x. That is, algorithm $\mathcal{A}$ runs on input x and returns y as output. When the algorithm $\mathcal{A}$ is deterministic, we write it as $y \leftarrow \mathcal{A}(x)$.

When X and Y are strings, we write $X||Y$ to mean the string created by appending Y to X. When $n > 0$ is an integer we write $\{0,1\}^n$ for the set of all n-bit strings.

We say a function $\mu(n)$ is negligible if $\mu \in o(n^{-\omega(1)})$, and is non-negligible otherwise. We let $\mathrm{negl}(n)$ denote an arbitrary negligible function. If we say some $p(n)$ is poly, we mean that there is some polynomial q such that for all sufficiently large n, $p(n) \le q(n)$. We say a function $\delta(n)$ is noticeable if the inverse $1/\delta(n)$ is poly.

2.1 Indifferentiability framework

In this subsection, we describe the indifferentiablity framework by Maurer et al [MRH04]. Our presentation here follows that by Ristenpart et al [RSS11]. Note that, the original version indifferentiablity framework by Maurer et al [MRH04] is based on random systems [Mau02]; later Coron et al present an alternative version [CDMP05] using interactive Turing machines. The formulation here we borrow from Ristenpart et al [RSS11] uses the game playing technique [BR06].

2.1.1 Game playing technique.

We use the game playing technique [BR06] as described in [RSS11]. Games consist of procedures which in turn consist of a sequence of statements together with some input and zero or more outputs. Procedures can call other procedures. If procedures P_1 and P_2 have inputs and outputs that are identical in number and type, we say that they export the same interface. If a procedure P gets access to procedure $\mathcal{F}$ we denote this by adding it in superscript $P^{\mathcal{F}}$. All variables used by procedures are assumed to be of local scope. After the execution of a procedure the variable values are left as they were after the execution of the last statement. If procedures are called multiple times, this allows them to keep track of their state.

A functionality $\mathcal{F}$ is a collection of two procedures $\mathcal{F}.hon$ and $\mathcal{F}.adv$, with suggestive names "honest" and "adversarial". Adversaries access a functionality $\mathcal{F}$ via the interface(s) exported by $\mathcal{F}.adv$, while all other procedures access the functionality via the interface(s) $\mathcal{F}.hon$.

Functionalities and games. Collections of procedures will sometimes implement particular abstract functionalities, for example that of some idealized primitive (e.g. a random oracle). A functionality is a collection $\mathcal{F} = (\mathcal{F}.hon, \mathcal{F}.adv)$; the names of these interfaces, hon and adv are suggestive as we will see in a moment. When games and adversaries are given access to a functionality a model of computation is defined. For example when the functionality is that of a random oracle, we have the random-oracle model. Thus one can think of functionalities and models somewhat interchangeably. As an example, functionality $\mathbf{RO} = (\mathbf{RO}.hon, \mathbf{RO}.adv)$, shown in Figure 2, implements a random oracle (with hon and adv interfaces) and will give rise to the random-oracle model.

```
procedure $\mathbf{RO}.hon(x)$:
If $\mathbb{T}[x] = \perp$ then $\mathbb{T}[x] \twoheadleftarrow \mathcal{R}$;
return $\mathbb{T}[x]$.

procedure $\mathbf{RO}.adv(x)$:
return $\mathbf{RO}.hon(x)$.
```

Figure 2: Procedures implementing the functionality of the random oracle model (ROM). The functionality is associated with randomness space $\mathcal{R} = \{0,1\}^r$ where the number $r \in \mathbb{N}$ is set as appropriate for a given context.

Similarly, functionality $\mathbf{RP} = (\mathbf{RP}.hon, \mathbf{RP}.adv)$, shown in Figure 3, implements a random permutation and will give rise to the random-permutation model. Note that, two "adversarial" interfaces, $\mathcal{P}.adv$ and $\mathcal{P}^{-1}.adv$ for capturing the permutation and the inverse, respectively, are defined so that the adversaries can access the functionality; and two "honest" interfaces, $\mathcal{P}.hon$ and $\mathcal{P}^{-1}.hon$ are defined for all other procedures to access the functionality.

```
procedure RP.hon:
INTERFACE $\mathcal{P}.hon(x)$:
If $\mathbb{T}[x] = \bot$ then $\mathbb{T}[x] \twoheadleftarrow \mathcal{R}$;
return $\mathbb{T}[x]$.

INTERFACE $\mathcal{P}^{-1}.hon(y)$:
If $\exists x$ so that $\mathbb{T}[x] = y$
    then $\mathbb{T}[x, y] \leftarrow x$;
return $\mathbb{T}[x, y]$.
```

```
procedure RP.adv:
INTERFACE $\mathcal{P}.adv(x)$:
return $\mathcal{P}.hon(x)$.

INTERFACE $\mathcal{P}^{-1}.hon(y)$:
return $\mathcal{P}^{-1}.hon(y)$.
```

Figure 3: Procedures implementing the functionality of the random permutation model (RPM). The functionality is associated with randomness space $\mathcal{R} = \{0, 1\}^r$ where the number $r \in \mathbb{N}$ is set as appropriate for a given context.

For any two functionalities $\mathcal{F}_1, \mathcal{F}_2$, we denote by $(\mathcal{F}_1, \mathcal{F}_2)$ the functionality that exposes a procedure that allows querying $(\mathcal{F}_1.hon, \mathcal{F}_2.hon)$ and a procedure that gives access to $(\mathcal{F}_1.adv, \mathcal{F}_2.adv)$.

A game $\mathcal{G}$ consists of a distinguished procedure called **main** (which takes no input) together with a set of procedures. A game can make use of functionality $\mathcal{F}$ and adversarial procedures $\mathcal{A}$ (together called "the adversary"). Adversarial procedures have access to the adversarial interface of functional procedures and, as any other procedure, can be called multiple times. We, however, restrict access to adversarial procedures to the game's main procedure, i.e., only it can call adversarial procedures and, in particular, adversarial procedures cannot call one another directly.

By $\mathcal{G}^{\mathcal{F},\mathcal{A}}$ we denote a game using functionality $\mathcal{F}$ and adversary $\mathcal{A}$. If $\mathcal{F}'$ exports the same interface as $\mathcal{F}$, and adversary $\mathcal{A}'$ exports the same interface as $\mathcal{A}$, then $\mathcal{G}^{\mathcal{F}',\mathcal{A}'}$ executes the same game $\mathcal{G}$ with functional procedure $\mathcal{F}'$ and adversary $\mathcal{A}'$. We denote by $\mathcal{G}^{\mathcal{F},\mathcal{A}} \Rightarrow y$ the event that game $\mathcal{G}$ produces output y, that is procedure main returns value y. If game $\mathcal{G}$ uses any probabilistic procedure then $\mathcal{G}^{\mathcal{F},\mathcal{A}}$ is a random variable and by $\Pr[\mathcal{G}^{\mathcal{F},\mathcal{A}} \Rightarrow y]$ we denote the probability (over the combined randomness space of the game) that it takes on value y. Sometimes we need to make the random coins r explicit and write $\mathcal{G}^{\mathcal{F},\mathcal{A}}(r)$ to denote that the game is run on random coins r.

Games are random variables over the entire random coins of the game and the adversarial procedures. For functionalities $\mathcal{F}$ and $\mathcal{F}'$ and adversaries $\mathcal{A}$ and $\mathcal{A}'$, we can thus consider the distance between the two random variables. Our security approach is that of concrete security, i.e., we say two games are ϵ-close if for all values y it holds that

$$\Pr\left[\mathcal{G}^{\mathcal{F},\mathcal{A}} \Rightarrow y\right] \leq \Pr\left[\mathcal{G}^{\mathcal{F}',\mathcal{A}'} \Rightarrow y\right] + \epsilon.$$

2.1.2 Indifferentiability.

Fix two functionalities $\mathcal{F}_1$ and $\mathcal{F}_2$. A distinguisher $\mathcal{D}$ is an adversary that outputs a bit. A simulator is a procedure, usually denoted $\mathcal{S}$. Figure 4 defines two games **Real** and **Ideal**. Fix some value y (e.g., $y = 1$). The indifferentiability advantage of $\mathcal{D}$ is defined as

$$\mathsf{Adv}^{\text{indiff}}_{\mathcal{F}_1,\mathcal{F}_2,\mathcal{S}}(\mathcal{D}) = \Pr[\mathbf{Real}^{\mathcal{F}_1,\mathcal{D}} \Rightarrow y] - \Pr[\mathbf{Ideal}^{\mathcal{F}_2,\mathcal{D}}_{\mathcal{S}} \Rightarrow y].$$

We use a concrete security approach, i.e. not providing a strict definition of achieving indifferentiability. However, informally we will say that a functionality $\mathcal{F}_1$ is indifferentiable from a functionality $\mathcal{F}_2$ if for any "reasonable" adversary $\mathcal{D}$ there exists an "efficient" simulator $\mathcal{S}$ such that $\mathsf{Adv}^{\text{indiff}}_{\mathcal{F}_1,\mathcal{F}_2,\mathcal{S}}(\mathcal{D})$ is "small". The meanings of "reasonable", "efficient", and "small" will be clear from context.

2.1.3 Composition.

One goal of indifferentiability is to allow the security analysis of a cryptographic scheme when using one functionality to imply security holds when using another. This is enabled by the following, which is a concrete security version of the original composition theorem of Maurer, Renner, and Holenstein [MRH04].

```
main Real:
    $b' \twoheadleftarrow \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$;
    return $b'$.

procedure $\mathsf{Func}(m)$:
    return $\mathcal{F}_1.hon(m)$.

procedure $\mathsf{Prim}(u)$:
    return $\mathcal{F}_1.adv(u)$.
```

```
main Ideal$_\mathcal{S}$:
    $b' \twoheadleftarrow \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$;
    return $b'$.

procedure $\mathsf{Func}(m)$:
    return $\mathcal{F}_2.hon(m)$.

procedure $\mathsf{Prim}(u)$:
    return $\mathcal{S}^{\mathcal{F}_2.adv}(u)$.
```

Figure 4: The games that define indifferentiability. Adversary $\mathcal{D}$ and functionalities $\mathcal{F}_1$, $\mathcal{F}_2$ are unspecified. The simulator $\mathcal{S}$ is a parameter of the game.

Theorem 2.1 *Let $\mathcal{F}_1$, $\mathcal{F}_2$ be two functionalities with compatible honest interfaces. Let $\mathcal{A}$ be an adversary with one oracle. Let $\mathcal{S}$ be a simulator that exports the same interface as $\mathcal{F}_1.adv$. Then there exist adversary $\mathcal{B}$ and distinguisher $\mathcal{D}$ such that for all values y*

$$\Pr[\mathcal{G}^{\mathcal{F}_1,\mathcal{A}} \Rightarrow y] \leq \Pr[\mathcal{G}^{\mathcal{F}_2,\mathcal{B}} \Rightarrow y] + \mathsf{Adv}^{\mathrm{indiff}}_{\mathcal{F}_1,\mathcal{F}_2,\mathcal{S}}(\mathcal{D}).$$

Moreover

$$t_\mathcal{B} \leq t_\mathcal{A} + q_\mathcal{A} \cdot t_\mathcal{S} \qquad q_\mathcal{B} \leq q_\mathcal{A} \cdot q_\mathcal{S} \qquad t_\mathcal{D} \leq t_\mathcal{G} + q_{\mathcal{G},1} \cdot t_\mathcal{A} \qquad q_\mathcal{D} \leq q_{\mathcal{G},0} + q_{\mathcal{G},1} \cdot q_\mathcal{A}$$

where $t_\mathcal{A}$, $t_\mathcal{B}$, $t_\mathcal{D}$ are the maximum running times of $\mathcal{A}$, $\mathcal{B}$, $\mathcal{D}$; $q_\mathcal{A}$, $q_\mathcal{B}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution; and $q_{\mathcal{G},0}$, $q_{\mathcal{G},1}$ are the maximum number of queries made by $\mathcal{G}$ to the honest interface and to the adversarial procedure.

3 Ideal Signatures: Previous Efforts and New Definitions

3.1 Analyzing the ideal signature in [ZZ20]

In this section, we analyze the ideal signature, including definition and construction, in [ZZ20].

3.1.1 Ideal signatures by Zhandry and Zhang

First, let's recall the definition of ideal signature from Zhandry and Zhang:

Definition 3.1 (Ideal Signature [ZZ20]) *Let $\mathcal{SK}, \mathcal{PK}, \mathcal{M}, \mathbf{\Sigma}$ be the sets such that:*

1. $|\mathcal{SK}| \geq 2^{\omega(\log\lambda)}$, $|\mathcal{PK}| \geq 2^{\omega(\log\lambda)}$, *and* $|\mathbf{\Sigma}| \geq 2^{\omega(\log\lambda)}$;
2. $|\mathcal{SK}| \leq |\mathcal{PK}|$, $|\mathcal{SK}| \times |\mathcal{M}| \leq |\mathbf{\Sigma}|$.

We denote

1. $\mathbf{G}[\mathcal{SK} \to \mathcal{PK}]$ *as the set of all injection that map $\mathcal{SK}$ to $\mathcal{PK}$;*
2. $\mathbf{S}[\mathcal{SK} \times \mathcal{M} \to \mathbf{\Sigma}]$ *as the set of all injections that map $\mathcal{SK} \times \mathcal{M}$ to $\mathbf{\Sigma}$;*
3. $\mathbf{V}[\mathcal{PK} \times \mathcal{M} \times \mathbf{\Sigma} \to \{0,1\}]$ *as the set of all functions that map $\mathcal{PK} \times \mathcal{M} \times \mathbf{\Sigma}$ to a bit.*

We define $\mathbf{T}$ as the set of all function tuples $(\textsc{Gen}, \textsc{Sign}, \textsc{Verify})$ such that:

- $\textsc{Gen} \in \mathbf{G}$, $\textsc{Sign} \in \mathbf{S}$ *and* $\textsc{Verify} \in \mathbf{V}$;
- $\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}$, $\textsc{Verify}(\textsc{Gen}(SK), M, \textsc{Sign}(SK, M)) = 1$;

- *$\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}, V \in \mathbf{\Sigma}$, if $V \neq \textsc{Sign}(SK, M)$, then $\textsc{Verify}(\textsc{Gen}(SK), M, \textsc{Sign}(SK, M)) = 0$;*
- *$\forall\ SK \in \mathcal{SK}, M \in \mathcal{M}, V_1, V_2 \in \mathbf{\Sigma}$, if $\textsc{Verify}(\textsc{Gen}(SK), M, V_1) = \textsc{Verify}(\textsc{Gen}(SK), M, V_2) = 1$, then $V_1 = V_2$.*

We say that a digital signature scheme $\Pi = \Pi.\{\textsc{Gen}, \textsc{Sign}, \textsc{Verify}\}$, associated with signing key space $\mathcal{SK}$, verification key space $\mathcal{PK}$, message space $\mathcal{M}$, and signature space $\mathbf{\Sigma}$, is an ideal digital signature, if Π is sampled from $\mathbf{T}$ uniformly.

3.1.2 Security analysis

Unfortunately, as already mentioned in the Introduction, the above definition in [ZZ20] is *not complete*. Specifically, let $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ be an ideal digital signature, associated with secret key space $\mathcal{SK}$, public key space $\mathcal{PK}$, and signature space $\mathbf{\Sigma}$. We define that a public key $PK \in \mathcal{PK}$ is honest, if there exists a secret key $SK \in \mathcal{SK}$ such that $\mathbf{Gen}(SK) = PK$, otherwise we say PK is dishonest. We immediately observe that, the verification algorithm $\mathbf{Verify}(PK, \cdot, \cdot)$ is not defined, where PK is dishonest, and thus we claim that this definition is not complete.

One would argue that this "incompleteness" can be trivially fixed by defining:

- $\forall PK \in \mathcal{PK}, M \in \mathcal{M}, V \in \mathbf{\Sigma}$, if $\nexists SK \in \mathcal{SK}$ such that $PK = \mathbf{Gen}(SK)$ and $V = \mathbf{Sign}(SK, M)$, then $\mathbf{Verify}(\textsc{Gen}(SK), M, \textsc{Sign}(SK, M)) = 0$.

Note that, in this definition, if PK is dishonest, then for any message M and signature value V, we have that $\mathbf{Verify}(PK, M, V) = 0$. We justify that this definition is natural, as in signature, we indeed wish that only the honest public key, along with the corresponding message and signature, could pass the verification test.

However, once completing the definition, the real problem comes. In the following, we illustrate a distinguishing attack that breaks the construction in [ZZ20]. Recalling their construction $\Pi = (\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify})$:

- $\Pi.\textsc{Gen}(SK) = \mathcal{P}(\Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}(\mathcal{H}_{\text{sk}}(SK)))$;
- $\Pi.\textsc{Sign}(SK, M) = \mathcal{E}(\Pi.\textsc{Gen}(SK)||M, \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}(\mathcal{H}_{\text{sk}}(SK), \mathcal{H}_{\text{msg}}(M)))$;
- $\Pi.\textsc{Verify}(PK, M, V) = \Pi_{\mathsf{sm}}.\textsc{Verify}_{\mathsf{sm}}(\mathcal{P}^{-1}(PK), \mathcal{H}_{\text{msg}}(M), \mathcal{E}^{-1}(PK||M, V))$,

where $\mathcal{H}_{\text{sk}}$ and $\mathcal{H}_{\text{msg}}$ are two random oracle models, $\mathcal{P}$ is a random permutation, $\mathcal{E}$ is an ideal cipher model. $\Pi_{\mathsf{sm}} = (\Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}, \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}, \Pi_{\mathsf{sm}}.\textsc{Verify}_{\mathsf{sm}})$ is a standard-model signature scheme satisfies: 1) uniqueness; 2) pseudorandom public keys; 3) random-message-attack security. To describe our distinguishing attack, we first build an alternative signature $\Pi_{\mathsf{sm\text{-}magic}} = (\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Gen}_{\mathsf{sm\text{-}magic}}, \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Sign}_{\mathsf{sm\text{-}magic}}, \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm\text{-}magic}})$ in the standard model. Concretely, let $\mathcal{PK}_{\mathsf{sm}}/\mathcal{SK}_{\mathsf{sm}}$ be the public/secret key space of Π_{sm}, and $pk_{\mathsf{sm\text{-}magic}}$ is a magic public key such that $pk_{\mathsf{sm\text{-}magic}} \notin \mathcal{PK}_{\mathsf{sm}}$, then we define the public/secret key space ($\mathcal{PK}_{\mathsf{sm\text{-}magic}}/\mathcal{SK}_{\mathsf{sm\text{-}magic}}$) of $\Pi_{\mathsf{sm\text{-}magic}}$ as

$$\mathcal{PK}_{\mathsf{sm\text{-}magic}} := \mathcal{PK}_{\mathsf{sm}} \cup \{pk_{\mathsf{sm\text{-}magic}}\}; \mathcal{SK}_{\mathsf{sm\text{-}magic}} := \mathcal{SK}_{\mathsf{sm}}.$$

Then we define $\Pi_{\mathsf{sm\text{-}magic}}$ as:

- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Gen}_{\mathsf{sm\text{-}magic}}\ (SK) = \Pi_{\mathsf{sm}}.\textsc{Gen}_{\mathsf{sm}}\ (SK)$;
- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Sign}_{\mathsf{sm\text{-}magic}}\ (SK, M) = \Pi_{\mathsf{sm}}.\textsc{Sign}_{\mathsf{sm}}\ (SK, M)$;
- $\Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm\text{-}magic}}\ (PK, M, V) = \begin{cases} \Pi_{\mathsf{sm\text{-}magic}}.\textsc{Verify}_{\mathsf{sm}}(PK, M, V) & \text{if } PK \neq pk_{\mathsf{sm\text{-}magic}}, \\ 1 & \text{if } PK = pk_{\mathsf{sm\text{-}magic}}. \end{cases}$

Trivial to note that $\Pi_{\mathsf{sm\text{-}magic}}$ also satisfies uniqueness, pseudorandom public keys, and random-message attack security, thus according to [ZZ20], we have that the signature construction, $\Pi_{\mathsf{magic}} = (\Pi_{\mathsf{magic}}.\textsc{Gen}, \Pi_{\mathsf{magic}}.\textsc{Sign}, \Pi_{\mathsf{magic}}.\textsc{Verify})$, that uses $\Pi_{\mathsf{sm\text{-}magic}}$ as a building block is also indifferentiable from an ideal signature. The differentiator who can distinguish the Π_{magic} from the ideal signature can be found in Figure 1 in the Introduction.

Note that, in the real world, the differentiator always outputs 1, due to the definition of $pk_{\mathsf{sm\text{-}magic}}$. However in the ideal world, if the simulator responds to the query with a honest public key (same as the proof in [ZZ20]), then the differentiator would output 0 with overwhelming probability, as the message M and signature value V are randomly sampled; and if the simulator responds to the query with a *dishonest* public key, then by definition, we have that differentiator would always output 0.

3.2 Our new definitions for ideal signatures and ideal one-time signatures

In this section, we define the functionality for ideal signature and for ideal one-time signature. We note that, the definitions for ideal signature and for ideal one-time signature are the first (complete) formulations in the indifferentability framework.

3.2.1 Our formulation of ideal signatures

As we argued in the previous subsection, the definition in [ZZ20] is not complete. In addition, in their the formulation in [ZZ20], Zhandry and Zhang intend to capture the uniqueness property. Our goal here is to capture the idealized version of ordinary signatures which can be randomized.

In Figure 5, we present the functionality $\mathbf{SIG} = (\mathbf{SIG}.hon, \mathbf{SIG}.adv)$, which implements an ideal signature and will give rise to the ideal signature model. Our presentation follows the framework by Ristenpart et al [RSS11]. In particular, we follow the *query-response style* for defining the interfaces of the idealized primitives, and the unnecessary structures can be clearly eliminated.

```
procedure SIG.hon

INTERFACE Gen.hon(SK):
If $\mathbb{T}[SK] = \bot$
    then $PK \twoheadleftarrow \mathcal{PK}$; $\mathbb{T}[SK] \leftarrow PK$;
return $\mathbb{T}[SK]$.

INTERFACE Sign.hon(PK, SK, M, R):
If $\mathbb{T}[SK] = PK$ and $\mathbb{T}[PK, SK, M, R] = \bot$
    then $V \twoheadleftarrow \mathbf{\Sigma}$; $\mathbb{T}[PK, SK, M, R] \leftarrow V$;
return $\mathbb{T}[PK, SK, M, R]$.

INTERFACE Verify.hon(PK, M, V):
If $\exists SK, R$ so that
        $\mathbb{T}[SK] = PK$ and $\mathbb{T}[PK, SK, M, R] = V$
    then $\phi \leftarrow 1$; $\mathbb{T}[PK, M, V] \leftarrow \phi$;
return $\mathbb{T}[PK, M, V]$.
```

```
procedure SIG.adv

INTERFACE Gen.adv(SK):
return Gen.hon(SK).

INTERFACE Sign.adv(PK, SK, M, R):
return Sign.hon(PK, SK, M, R).

INTERFACE Verify.adv(PK, M, V):
return Verify.hon(PK, M, V).
```

Figure 5: Procedures implementing the functionality of the ideal signature model (ISM). The associated parameters are verification key space $\mathcal{PK}$, signing key space $\mathcal{SK}$, message space $\mathcal{M}$, randomness space $\mathcal{R}$, and signature space $\mathbf{\Sigma}$.

In the ideal signature in Figure 5, three "honest" interfaces, $\mathbf{Gen}.hon$, $\mathbf{Sign}.hon$, $\mathbf{Verify}.hon$, are defined, for capturing key generation, signing and verification, respectively. Here, "adversarial" interfaces are identical to the honest ones. Several tables $\mathbb{T}[]$ have been used to trace the behaviors of the ideal signature. Notation "$\mathbb{T}[x] \leftarrow y$" means that, the value of the x-the record in the table is y; equivalently, for the query value x, the (potential) response value is y. In the ideal signature in Figure 5, the response values for PK and for signature V are randomly sampled. Whenever a signature is generated, the involved signing key SK must be well-defined, and be aware to the ideal signature.

3.2.2 Our formulation for ideal one-time signatures

In Figure 6, we present the functionality $\mathbf{OSIG} = (\mathbf{OSIG}.hon, \mathbf{OSIG}.adv)$, which implements an ideal *one-time* signature and will give rise to the ideal one-time signature model. Similar to the ideal signature formulation, three "honest" interfaces, $\mathbf{oGen}.hon$, $\mathbf{oSign}.hon$, $\mathbf{oVerify}.hon$, are defined for capturing key generation, one-time signing, and signature verification, respectively. However, now the "adversarial" interfaces are *different* from the "honest" interfaces. Indeed, the "adversarial" interfaces of ideal one-time signature are *identical to the adversarial/honest interfaces of the ideal signature in Figure 5.*

```
procedure OSIG.hon

INTERFACE oGen.hon($SK$):
If $\mathbb{T}[SK] = \perp$
    then $PK \twoheadleftarrow \mathcal{PK}$; $\mathbb{T}[SK] \leftarrow PK$;
return $\mathbb{T}[SK]$.

INTERFACE oSign.hon($PK, SK, M, R$):
If $\mathbb{T}[SK] = PK$ and $\mathbb{T}[PK, SK, *, *] = \perp$
    then $V \twoheadleftarrow \mathbf{\Sigma}$; $\mathbb{T}[PK, SK, M, R] \leftarrow V$;
        $\mathbb{T}[PK, SK, M, R, V] \leftarrow 1$;
return $\mathbb{T}[PK, SK, M, R]$.

INTERFACE oVerify.hon($PK, M, V$):
If $\exists SK, R$ so that
        $\mathbb{T}[PK, SK, M, R, V] = 1$
    then $\phi \leftarrow 1$; $\mathbb{T}[PK, M, V] \leftarrow \phi$;
return $\mathbb{T}[PK, M, V]$.
```

```
procedure OSIG.adv

INTERFACE oGen.adv($SK$):
return oGen.hon($SK$).

INTERFACE oSign.adv($PK, SK, M, R$):
If $\mathbb{T}[SK] = PK$ and $\mathbb{T}[PK, SK, M, R] = \perp$
    then $V \twoheadleftarrow \mathbf{\Sigma}$; $\mathbb{T}[PK, SK, M, R] \leftarrow V$;
return $\mathbb{T}[PK, SK, M, R]$.

INTERFACE oVerify.adv($PK, M, V$):
If $\exists SK, R$ so that
        $\mathbb{T}[SK] = PK$ and $\mathbb{T}[SK, M, R] = V$
    then $\phi \leftarrow 1$; $\mathbb{T}[PK, M, V] \leftarrow \phi$;
return $\mathbb{T}[PK, M, V]$.
```

Figure 6: Procedures implementing the functionality of the ideal one-time signature model. The associated parameters are verification key space $\mathcal{PK}$, signing key space $\mathcal{SK}$, message space $\mathcal{M}$, randomness space $\mathcal{R}$, and signature space $\mathbf{\Sigma}$.

More concretely, through the "adversarial" interfaces of ideal one-time signature, the same signing key SK is allowed to be used for signing multiple distinct messages. In contrast, this is *not* allowed through the honest interfaces of ideal one-time signature. Now in Figure 6, the condition "$\mathbb{T}[SK] = PK$ and $\mathbb{T}[PK, SK, *, *] = \perp$" enforces that, when SK has never been used for signing any message, a new fresh signature V will be sampled. Additional records $\mathbb{T}[PK, SK, M, R, V] \leftarrow 1$ are included, to make sure of effective verification through the honest interfaces of ideal one-time signature.

4 Ideal One-time Signature from Random Oracle

4.1 Construction

High-level ideas. Our goal here is to construct a scheme which is indifferentiable from the ideal one-time signature (as defined in Figure 6). The starting point of our construction is Lamport's one-time signature scheme. In Lamport's designa one-way function f is used in the key generation algorithm; more concretely, the signing key $\boldsymbol{sk}$ consists of an n-by-2 matrix of random strings denoted as, $\boldsymbol{sk} = \langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle$, while the corresponding verification key $\boldsymbol{pk}$ also consists of an n-by-2 matrix of strings, i.e., $\boldsymbol{pk} = \langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle$, where $pk_{i,0} = f(sk_{i,0})$ and $pk_{i,1} = f(sk_{i,1})$ for all $1 \leq i \leq n$. A collision resistant hash function H is used in the signing algorithm; to sign a given message m, first the message is compressed into a n-bit string $b_1 b_2 \cdots b_n \leftarrow H(\boldsymbol{pk}, m)$, where $b_i \in \{0, 1\}$; the corresponding signature $\boldsymbol{\sigma}$ consists of "half of the signing key", i.e., $\boldsymbol{\sigma} = sk_{1,b_1} || \cdots || sk_{n,b_n}$. Finally, In the verification algorithm, upon receiving a message m and signature $\boldsymbol{\sigma} = \sigma_1 || \cdots || \sigma_n$, first compute $b_1 b_2 \cdots b_n \leftarrow H(\boldsymbol{pk}, m)$; if it holds that, $f(\sigma_i) = pk_{i,b_i}$, for all $i = 1, \ldots, n$, then the message-signature pair $(m, \boldsymbol{\sigma})$ is valid.

The trivial idea to build an indifferentiable digital signature is to apply the construction in [ZZ20], by setting the standard model signature scheme to be Lamport's signature. However, as show in Section 3.1, this idea is unknown to be sufficient, due to the dishonest public key attack. To prevent this attack, we eliminate the dishonest public key by defining **oGen** (the key generating algorithm of an ideal one-time signature) to be a random function and SK to be large enough to make sure that every $PK \in \mathcal{PK}$ is honest with high probability. We then prove that, after combining this new technique, our construction is indifferentiable from an ideal one-time signature. Here, we first specify some parameters:

- $\lambda^{O(1)} \geq n \geq \omega(\log \lambda)$,

- $\log|\mathcal{SK}_\circ| = 8n\lambda, \log|\mathcal{PK}_\circ| = 2n\lambda$, $\log|\mathbf{\Sigma}_\circ| = 2n\lambda$.

Building blocks. Based on the above discussion, in our design we will use the following building blocks:

- $\mathcal{H}_{\rm sk} : \{0,1\}^* \to \{0,1\}^{4n\lambda}$ is a random oracle.
- $\mathcal{H}_{\rm OneWay} : \{0,1\}^* \to \{0,1\}^{\lambda}$ is a random oracle.
- $\mathcal{H}_{\rm position} : \{0,1\}^* \to \{0,1\}^{n}$ is a random oracle.
- $\mathcal{P} : \{0,1\}^{2n\lambda} \to \{0,1\}^{2n\lambda}$ is an ideal permutation and $\mathcal{P}^{-1}$ is its inverse.
- $\mathcal{E} : \{0,1\}^{\log|\mathcal{PK}_\circ|+\log|\mathcal{M}_\circ|} \times \{0,1\}^{\log|\mathbf{\Sigma}_\circ|} \to \{0,1\}^{\log|\mathbf{\Sigma}_\circ|}$ is an ideal cipher model, where $\{0,1\}^{\log|\mathcal{PK}_\circ|+\log|\mathcal{M}_\circ|}$ is its key space and $\mathcal{E}^{-1}$ is its inverse. That is, $\mathcal{E}^{-1} : \{0,1\}^{\log|\mathcal{PK}_\circ|+\log|\mathcal{M}_\circ|} \times \{0,1\}^{\log|\mathbf{\Sigma}_\circ|} \to \{0,1\}^{\log|\mathbf{\Sigma}_\circ|}$.

4.1.1 Construction details.

Now we are ready to build the indifferentiable one-time signature, denoted as $\Pi_\circ = (\Pi_\circ.\text{oGen}, \Pi.\text{oSign}, \Pi.\text{oVerify})$.

$\Pi_\circ = (\Pi_\circ.\textbf{oGen}, \Pi_\circ.\textbf{oSign}, \Pi_\circ.\textbf{oVerify})$

```
PK_∘ ← Π_∘.oGen(SK_∘):
On input, signing key $SK_\circ$, compute the verification key $PK_\circ$, as follows:
  1. $\boldsymbol{sk} \leftarrow \mathcal{H}_{\rm sk}(SK_\circ)$; parse $\boldsymbol{sk}$ into $\big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\big)$;
  2. for $i \in [1,n]$,
         $pk_{i,0} \leftarrow \mathcal{H}_{\rm OneWay}(sk_{i,0})$; $pk_{i,1} \leftarrow \mathcal{H}_{\rm OneWay}(sk_{i,1})$;
     $\boldsymbol{pk} \leftarrow \big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\big)$;
  3. $PK_\circ \leftarrow \mathcal{P}(\boldsymbol{pk})$; return $PK_\circ$.

$V_\circ \leftarrow \Pi_\circ.\text{oSign}(PK_\circ, SK_\circ, M_\circ)$:
On input, verification key $PK_\circ$, signing key $SK_\circ$, message $M_\circ$, compute the signature $V_\circ$, as follows:
  1. If $PK_\circ = \text{oGen}(SK_\circ)$
         then $\boldsymbol{sk} \leftarrow \mathcal{H}_{\rm sk}(SK_\circ)$; parse $\boldsymbol{sk}$ into $\big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\big)$;
              $b_1 \cdots b_n \leftarrow \mathcal{H}_{\rm position}(PK_\circ, M_\circ)$;
  2. output the signature
         $V_\circ \leftarrow \mathcal{E}\big(PK_\circ || M_\circ,\ sk_{1,b_1} || \cdots || sk_{n,b_n}\big)$.

$\phi \leftarrow \Pi_\circ.\text{oVerify}(PK_\circ, M_\circ, V_\circ)$:
On input, verification key $PK_\circ$, message $M_\circ$ and signature $V_\circ$, operate as follows:
  1. "unpack" the signature
         $\big(\widehat{sk}_{1,\widehat{b}_1} || \cdots \widehat{sk}_{n,\widehat{b}_n}\big) \leftarrow \mathcal{E}^{-1}(PK_\circ || M_\circ,\ V_\circ)$;
  2. "unpack" the verification key
         $\big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\big) \leftarrow \mathcal{P}^{-1}(PK_\circ)$;
  3. $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\rm position}(PK_\circ || M_\circ)$;
  4. output $\phi \leftarrow 1$ if and only if the following conditions hold:
       • for $k \in [1,n]$,
             $\mathcal{H}_{\rm OneWay}(\widehat{sk}_{i,\widehat{b}_i}) = \widehat{pk}_{i,\widehat{b}_i}$.
     otherwise output $\phi \leftarrow 0$.
```

4.2 Security statement

The construction in previous subsection is an ideal one-time signature. More concretely, we have the following theorem.

Theorem 4.1 $\Pi_\circ = (\Pi_\circ.\textit{oGen}, \Pi.\textit{oSign}, \Pi.\textit{oVerify})$ *is indifferentiable from the ideal one-time signature* $\mathbf{OSIG} = (\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$, *in the model for, random oracles* $\mathcal{H}_{\mathrm{sk}}$, $\mathcal{H}_{\mathrm{OneWay}}$, $\mathcal{H}_{\mathrm{position}}$, *random permutation* $(\mathcal{P}, \mathcal{P}^{-1})$, *and ideal cipher* $(\mathcal{E}, \mathcal{E}^{-1})$. *More precisely, there exists a simulator* $\mathcal{S}$ *such that for all* q*-query differentiator* $\mathcal{D}$, *we have*

$$\mathsf{Adv}^{\mathsf{indiff}}_{\Pi_\circ,\mathcal{S},\mathcal{D}} \leq \frac{q^2}{2^n} + \frac{2q^2}{2^\lambda}.$$

Here, the simulator makes at most q^2 *queries to its oracles.*

4.3 The simulator and high-level proof ideas

According to the definition of indifferentiability, we have that, in the real world, the differentiator has three honest interfaces ($\Pi_\circ$.oGen, $\Pi_\circ$.oSign, $\Pi_\circ$.oVerify) and seven adversarial interfaces ($\mathcal{H}_{\mathrm{sk}}$, $\mathcal{H}_{\mathrm{OneWay}}$, $\mathcal{H}_{\mathrm{position}}$, $\mathcal{P}$, $\mathcal{P}^{-1}$, $\mathcal{E}$, $\mathcal{E}^{-1}$). Therefore, to complete the proof, we build an efficient simulator $\mathcal{S}$ in the ideal world, such that 1) $\mathcal{S}$ has access to the ideal one-time signature via the adversarial interfaces; 2) $\mathcal{S}$ simulates those seven adversarial interfaces properly. Concretely, in the ideal world, the differentiator $\mathcal{D}$ has three honest interfaces ($\mathbf{oGen}$, $\mathbf{oSign}$, $\mathbf{oVerify}$) and seven adversarial interfaces ($\mathcal{S}^{\mathcal{H}_{\mathrm{sk}}}$, $\mathcal{S}^{\mathcal{H}_{\mathrm{OneWay}}}$,$\mathcal{S}^{\mathcal{H}_{\mathrm{position}}}$, $\mathcal{S}^{\mathcal{P}}$, $\mathcal{S}^{\mathcal{P}^{-1}}$ $\mathcal{S}^{\mathcal{E}}$, $\mathcal{S}^{\mathcal{E}^{-1}}$), and we prove that for any differentiator $\mathcal{D}$, the view in the real world is close to the view in the ideal world. In the following, we illustrate the full description of our simulator and then we give the high-level intuition of our proof strategy.

Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ has the external oracle access to the ideal one-time signature $\mathbf{OSIG}$ =($\mathbf{oGen}$, $\mathbf{oSign}$, $\mathbf{oVerify}$); the simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}^{\mathcal{H}_{\mathrm{sk}}}(SK_\circ)$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{sk}$;
query the external $\mathbf{OSIG}$ with ($\mathbf{oGen}$, $SK_\circ$), and obtain $\widehat{PK_\circ}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$ s.t. $PK_\circ = \widehat{PK_\circ}$
    then return $\boldsymbol{sk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$; $\boldsymbol{pk} \twoheadleftarrow \{0,1\}^{2n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\mathcal{S}^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk})$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \twoheadleftarrow \{0,1\}^\lambda$; $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\mathcal{S}^{\mathcal{H}_{\mathrm{position}}}(PK_\circ, M_\circ)$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
```

```
$b_1 \cdots b_n \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}^{\mathcal{P}}(\boldsymbol{pk})\text{:}}$

if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;

parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
for $i \in [1,n]$   //compute the corresponding secret key of $\boldsymbol{pk}$
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$;
        else $\widehat{sk}_{i,0} \leftarrow \{0,1\}^{2\lambda}$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$;
        else $\widehat{sk}_{i,1} \leftarrow \{0,1\}^{2\lambda}$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$;
$\boldsymbol{sk} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;

$SK_\circ \twoheadleftarrow \mathcal{SK}_\circ$; $PK_\circ \leftarrow \mathbf{oGen}(SK_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $PK_\circ$.

$\underline{\mathcal{S}^{\mathcal{P}^{-1}}(PK_\circ)\text{:}}$

if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$; $\boldsymbol{pk} \twoheadleftarrow \{0,1\}^{2n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n})\text{:}}$

if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;

$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then $b_1^* \cdots b_n^* \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1^* \cdots b_n^*)\}$; $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;

parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,   //check the validity of each $sk_{i,b_i}$
```

```
        then $ctr \leftarrow ctr + 1$;

    if $ctr < n$
        then goto Case 1;
        else goto Case 2;

    Case 1: $V_\circ \twoheadleftarrow \mathbf{\Sigma}_\circ; \mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})\}$; return $V_\circ$.
    Case 2: query the external $\mathbf{OSIG}$ with $(\mathbf{oSign}, \widehat{SK_\circ}, M_\circ)$, and obtain $V_\circ$;
            $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})\}$; return $V_\circ$.

$\underline{\mathcal{S}^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)}$:
    if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}) \in \mathbb{T}_\mathcal{E}$,
        then return $(sk_{1,b_1}, \ldots, sk_{n,b_n})$.

    if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
        then return $(sk_{1,b_1}, \ldots, sk_{n,b_n})$.

    query the external $\mathbf{OSIG}$ with $(\mathbf{oVerify}, PK_\circ||M_\circ, V_\circ)$, and obtain $\widehat{\phi}$;
    if $\widehat{\phi} = 0$, //for the invalid signature, respond with random strings
        then for $i \in [1, n]$
                $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
            $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
            return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.

    if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
        then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
        else $\widehat{b}_1, \cdots \widehat{b}_n \twoheadleftarrow \{0,1\}^n$;
    if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
        then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
    if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
        then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
        else $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$; $\widehat{\boldsymbol{pk}} \twoheadleftarrow \{0,1\}^{2n\lambda}$;

    parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
    parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
    for $i \in [1, n]$
        $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
    $\widehat{pad} \leftarrow 0 \cdots 0$;
    $\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
    return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, )$.
```

We immediately note that, if the differentiator $\mathcal{D}$ makes q queries via the adversarial interfaces, then our simulator $\mathcal{S}$ makes at most q^2 queries to the ideal one-time signature $\mathbf{OSIG} = (\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$. Here, $\mathcal{S}$ only keeps nine tables with size at most $2nq$ ($\mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$), which means that the constructed simulator $\mathcal{S}$ is efficient. In the following, we present the intuitive ideas to show that why $\mathcal{S}$ works, and in Appendix A we give the detailed proof. Note that, in the real world construction, the building blocks $\mathcal{H}_{\text{sk}}$, $\mathcal{H}_{\text{OneWay}}$, and $\mathcal{H}_{\text{position}}$, are random oracles, $\mathcal{P}$ is an ideal permutation associated with its inverse $\mathcal{P}^{-1}$, $\mathcal{E}$ is an ideal cipher associated with its inverse $\mathcal{E}^{-1}$. Hence, the responses of a proper simulator must follow the following rules:

1. The responses of $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$, $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}$, and $\mathcal{S}^{\mathcal{H}_{\text{position}}}$ are statistically close to the uniform;
2. The responses of $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$, $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}$, and $\mathcal{S}^{\mathcal{H}_{\text{position}}}$ are consistent;
3. The responses of $(\mathcal{S}^{\mathcal{P}}, \mathcal{S}^{\mathcal{P}^{-1}})$ are statistically close to those of a random permutation;
4. There exists no $(\boldsymbol{pk} \neq \boldsymbol{pk}')$, such that $\mathcal{P}(\boldsymbol{pk}) = \mathcal{P}(\boldsymbol{pk}')$;

5. There exists no $(PK_{\circ} \neq PK'_{\circ})$ such that $\mathcal{P}^{-1}(PK_{\circ}) = \mathcal{P}^{-1}(PK'_{\circ})$;

6. For fixed $(PK_{\circ}||M_{\circ})$, the responses of $(\mathcal{S}^{\mathcal{E}}, \mathcal{S}^{\mathcal{E}^{-1}})$ are statistically close to those of an ideal cipher;

7. There exists no $\Big((\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}) \neq (\widehat{sk}'_{1,b'_1}, \ldots, \widehat{sk}'_{n,b'_n})\Big)$ such that

$$\mathcal{S}^{\mathcal{E}}(PK_{\circ}||M_{\circ}, \widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}) = \mathcal{S}^{\mathcal{E}}(PK_{\circ}||M_{\circ}, \widehat{sk}'_{1,b'_1}, \ldots, \widehat{sk}'_{n,b'_n});$$

8. There exists no $(V_{\circ} \neq V'_{\circ})$ such that

$$\mathcal{S}^{\mathcal{E}^{-1}}(PK_{\circ}||M_{\circ}, V_{\circ}) = \mathcal{S}^{\mathcal{E}^{-1}}(PK_{\circ}||M_{\circ}, V'_{\circ});$$

9. $\mathbf{oGen}(SK_{\circ}) = \mathcal{S}^{\Pi_{\circ}.\text{oGEN}}(SK_{\circ})$;

10. $\mathbf{oSign}(SK_{\circ}, M_{\circ}) = \mathcal{S}^{\Pi_{\circ}.\text{oSIGN}}(SK_{\circ}, M_{\circ})$;

11. $\mathbf{oVerify}(PK_{\circ}, M_{\circ}, V_{\circ}) = \mathcal{S}^{\Pi_{\circ}.\text{oVERIFY}}(PK_{\circ}, M_{\circ}, V_{\circ})$.

Next, we illustrate why and how $\mathcal{S}$ achieves these eleven rules.

Rule 1. We here show that the responses of $\mathcal{S}^{\mathcal{H}_{\text{sk}}}, \mathcal{S}^{\mathcal{H}_{\text{OneWay}}}, \mathcal{S}^{\mathcal{H}_{\text{position}}}$ are well distributed.

The responses of $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$. By definition, $\mathcal{S}$ responds to the query $SK_{\circ}$, i.e., $\mathcal{S}^{\mathcal{H}_{\text{sk}}}(SK_{\circ})$, by either using the table $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$ or returning a uniformly sampled sk. Concretely,

- If there is a tuple $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ}) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$, then $\mathcal{S}$ responds with $\boldsymbol{sk}$;
- If there is a tuple $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ}) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ such that $\mathbf{oGen}(SK_{\circ}) = PK_{\circ}$, then $\mathcal{S}$ responds with $\boldsymbol{sk}$;
- Else, $\mathcal{S}$ responds with a uniformly sampled $\boldsymbol{sk}$.

It is trivial to note that, the response of the last case is well-distributed. Next, we analyze the first two cases. For the first case, we note that the tuple $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ is inserted into $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$ in two ways, by $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$ or by $\mathcal{S}^{\mathcal{P}}$: if the tuple is inserted by $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$, then the response is fine as $\boldsymbol{sk}$ is uniformly sampled, while if the tuple is inserted by $\mathcal{S}^{\mathcal{P}}$, then the response might not be uniformly distributed, as $\boldsymbol{sk}$ can be chosen by the adversary. Fortunately, we observe that if the tuple $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ is inserted by $\mathcal{S}^{\mathcal{P}}$, then $SK_{\circ}$ is uniformly sampled by $\mathcal{S}$ which is independent of the differentiator's view. Thus, the differentiator $\mathcal{D}$ would not make such a query except for negligible probability $(\leq \frac{q}{|\mathcal{SK}_{\circ}|} = \frac{q}{2^{8n\lambda}})$. For the second case, the tuple $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ is inserted either by $\mathcal{S}^{\mathcal{P}^{-1}}$ or by $\mathcal{S}^{\mathcal{E}^{-1}}$; note that here $\boldsymbol{sk}$ is uniformly sampled in $\mathcal{S}^{\mathcal{P}^{-1}}$ or in $\mathcal{S}^{\mathcal{E}^{-1}}$, which means that the response is well-distributed. However, if the differentiator $\mathcal{D}$ has $(SK_{\circ} \neq SK'_{\circ})$ such that $\mathbf{oGen}(SK_{\circ}) = \mathbf{oGen}(SK'_{\circ}) = PK_{\circ}$, then $\mathcal{S}^{\mathcal{H}_{\text{sk}}}(SK_{\circ}) = \mathcal{S}^{\mathcal{H}_{\text{sk}}}(SK'_{\circ})$, which induces a collision. While, due to the definition of ideal one-time signature, this bad event would not occur except for negligible probability $(\leq \frac{q^2}{|\mathcal{PK}_{\circ}|} = \frac{q^2}{2^{2n\lambda}})$.

The responses of $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}$. By definition, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})$ by either using the table $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$ or returning a uniformly sampled $\widehat{pk}$. Concretely,

- If there is a pair $(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$, then $\mathcal{S}$ responds with $\widehat{pk}$;
- Else, $\mathcal{S}$ responds with a uniformly sampled $\widehat{pk}$.

It is trivial to observe that the response of the last one is well distributed. For the first one, the pair $(\widehat{sk}, \widehat{pk})$ might be inserted in four ways, by $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}, \mathcal{S}^{\mathcal{P}}, \mathcal{S}^{\mathcal{P}^{-1}}$, or $\mathcal{S}^{\mathcal{E}^{-1}}$. Note that if the pair is inserted by $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}, \mathcal{S}^{\mathcal{P}^{-1}}$, or $\mathcal{S}^{\mathcal{E}^{-1}}$, then $\widehat{pk}$ is uniformly sampled, which means that the response is well distributed. However, if the pair is inserted by $\mathcal{S}^{\mathcal{P}}$, then the response would be controlled by the adversary and fail to be uniform. Same as above, we have that the differentiator would not make such a query except for negligible probability ($\leq \frac{2nq}{2^{2\lambda}}$) as $\widehat{sk}$ is uniformly sampled by $\mathcal{S}$.

The response of $\mathcal{S}^{\mathcal{H}_{\text{position}}}$. By definition, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{H}_{\text{position}}}(PK_{\circ}, M_{\circ})$ by either using the tables $\mathbb{T}_{\mathcal{H}_{\text{position}}}, \mathbb{T}_{\mathcal{E}^{-1}}$ or returning a uniformly sampled n-bit string $b_1 \cdots b_n$. It is trivial to note that in all the cases, $b_1 \cdots b_n$ is uniformly sampled, which means that the response is well-distributed.

Rule 2. By definition, we immediately observe that the responses of $\mathcal{S}^{\mathcal{H}_{\text{position}}}$ are always consistent.

Consistency of $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$. Note that the only bad case that causes inconsistency for $\mathcal{S}^{\mathcal{H}_{\text{sk}}}(SK_{\circ})$ is that the table $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$ records two tuples: $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ and $(\diamond, \boldsymbol{sk}', \boldsymbol{pk}', PK_{\circ})$. Concretely, $\mathcal{S}^{\mathcal{P}^{-1}}$ first inserts $(\diamond, \boldsymbol{sk}', \boldsymbol{pk}', PK_{\circ})$ and then $\mathcal{S}^{\mathcal{P}}$ inserts $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$. Observe that, $SK_{\circ}$ is uniformly sampled by $\mathcal{S}$, which means that collision on $PK_{\circ}$ would not happen except for negligible probability ($\leq \frac{q^2}{|\mathcal{PK}_{\circ}|} = \frac{q^2}{2^{n\lambda}}$).

Consistency of $\mathcal{S}^{\mathcal{H}_{\text{OneWay}}}$. Note that if there is no collision on $\widehat{sk}$ ($\leq \frac{(2qn)^2}{2^{2\lambda}}$), then the consistency holds trivially.

Rule 3. For this rule, it suffices to show that for any pair $(\boldsymbol{pk}, PK_{\circ})$ such that $\mathcal{S}^{\mathcal{P}}(\boldsymbol{pk}) = PK_{\circ}$, where either $\boldsymbol{pk}$ or $PK_{\circ}$ is uniformly sampled. Here, for the query $\mathcal{S}^{\mathcal{P}}(\boldsymbol{pk})$, either $\boldsymbol{pk}$ is uniformly sampled by $\mathcal{S}$ ($\mathcal{S}^{\mathcal{P}^{-1}}$ inserts $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ into $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$) or $\mathcal{S}$ responds to the query with $\mathbf{oGen}(SK_{\circ})$ ($SK_{\circ} \twoheadleftarrow \mathcal{SK}_{\circ}$), which means the response is well distributed. Analogically, for the query $\mathcal{S}^{\mathcal{P}^{-1}}(PK_{\circ})$, either $PK_{\circ}$ is with form of $\mathbf{oGen}(SK_{\circ})$ ($\mathcal{S}^{\mathcal{P}}$ inserts $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$), referring to $PK_{\circ}$ is uniformly distributed, or the response $\boldsymbol{pk} = \mathcal{S}^{\mathcal{P}^{-1}}(PK_{\circ})$ is uniformly sampled by $\mathcal{S}^{\mathcal{P}^{-1}}$.

Rule 4. Note that there are two bad cases that breaks this rule,

- $\mathcal{S}^{\mathcal{P}^{-1}}$ first inserts $(\diamond, \boldsymbol{sk}', \boldsymbol{pk}', PK_{\circ})$ and then $\mathcal{S}^{\mathcal{P}}$ inserts $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$;
- $\mathcal{S}^{\mathcal{P}}$ first inserts $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ and then $\mathcal{S}^{\mathcal{P}}$ inserts $(SK_{\circ}, \boldsymbol{sk}', \boldsymbol{pk}', PK_{\circ})$.

The former one is trivially bounded by Rule 2 (consistency of $\mathcal{H}_{\text{sk}}$), and the latter one would only occur if there is a collision on $PK_{\circ}$ ($\frac{q^2}{|\mathcal{PK}_{\circ}|} = \frac{q^2}{2^{2n\lambda}}$). Specifically, for both of the queries $(\mathcal{S}^{\mathcal{P}}(\boldsymbol{pk}))$ and $(\mathcal{S}^{\mathcal{P}}(\boldsymbol{pk}'))$, $\mathcal{S}$ samples $SK_{\circ}$ and $SK'_{\circ}$ such that $\mathbf{oGen}(SK_{\circ}) = \mathbf{oGen}(SK'_{\circ}) = PK_{\circ}$, and responds with $PK_{\circ}$.

Rule 5. Similar to Rule 4, there are also two bad cases that would break this rule,

- $\mathcal{S}^{\mathcal{P}}$ or $\mathcal{S}^{\mathcal{H}_{\text{sk}}}$ first inserts $(SK_{\circ}, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ and then $\mathcal{S}^{\mathcal{P}^{-1}}$ inserts $(\diamond, \boldsymbol{sk}', \boldsymbol{pk}, PK'_{\circ})$[6];
- $\mathcal{S}^{\mathcal{P}^{-1}}$ first inserts $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\circ})$ and then $\mathcal{S}^{\mathcal{P}^{-1}}$ inserts $(\diamond, \boldsymbol{sk}', \boldsymbol{pk}, PK'_{\circ})$;

Immediately observe that the former one will never occur unless there is a collision on $\boldsymbol{pk}$ ($\leq \frac{q^2}{2^{2n\lambda}}$). Concretely, $\mathcal{S}$ uniformly samples the same $\boldsymbol{pk}$ and sets $\boldsymbol{pk} = \mathcal{S}^{\mathcal{P}^{-1}}(PK_{\circ})$. Within the similar analysis, the latter one would not happen except for a collision on $\boldsymbol{pk}$ ($\leq \frac{q^2}{2^{2n\lambda}}$). Specifically, for both of the queries $\mathcal{S}^{\mathcal{P}^{-1}}(PK_{\circ})$ and $\mathcal{S}^{\mathcal{P}^{-1}}(PK'_{\circ})$, $\mathcal{S}$ samples the same $\boldsymbol{pk}$ and sets $\boldsymbol{pk} = \mathcal{S}^{\mathcal{P}^{-1}}(PK_{\circ}) = \mathcal{S}^{\mathcal{P}^{-1}}(PK'_{\circ})$.

Rule 6. Similar to the analysis in Rule 3, it suffices to prove that for fixed $(PK_{\circ}||M_{\circ})$, any pair $\Big((sk_{1,b_1}, \ldots, sk_{n,b_n},), V_{\circ}\Big)$ such that

$$\mathcal{S}^{\mathcal{E}}(PK_{\circ}||M_{\circ}, sk_{1,b_1}, \ldots, sk_{n,b_n},) = V_{\circ},$$

either $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$ or $V_{\circ}$ are uniformly sampled. By the description of our simulator, note that if $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$ is not uniformly sampled (say, chosen by the differentiator), then $V_{\circ}$ is either with form of $\mathbf{oSign}(SK_{\circ}, M_{\circ})$ (Case 2 in the green-box above on page 17) or just a random string (Case 1 in the red-box above on page 17). Moreover, we note that, if $V_{\circ} = \mathbf{oSign}(SK_{\circ}, M_{\circ})$, then $V_{\circ}$ is uniformly distributed, under the condition that no $(SK_{\circ} \neq SK'_{\circ})$ such that $\mathbf{oGen}(SK_{\circ}) = \mathbf{oSign}(SK'_{\circ}) = PK_{\circ}$ (bounded by Rule 1). Thus, if $V_{\circ}$ is not uniformly distributed (say, chosen by the differentiator), then $\mathbf{oVerify}(PK_{\circ}||M_{\circ}, V_{\circ}) = 0$ except for negligible probability ($\leq \frac{|\mathcal{SK}_{\circ}|}{|\boldsymbol{\Sigma}_{\circ}|}$).

[6]Note, $\boldsymbol{sk}$ and $\boldsymbol{sk}'$ might be either identical or distinct and we don't care about it in this rule.

We immediately observe that if $\mathbf{oVerify}(PK_\circ || M_\circ, V_\circ) = 0$, then $\mathcal{S}^{\mathcal{E}^{-1}}$ would responds with uniformly sampled $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$.

Rule 7. Similar to Rule 4, there are three bad events that might break the rule,

- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}}$ inserts $(V_\circ, PK_\circ, M_\circ, sk'_{1,b_1}, \ldots, sk'_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
  and then $\mathcal{S}^{\mathcal{E}}$ inserts $(V_\circ, PK_\circ, M_\circ, sk'_{1,b_1}, \ldots, sk'_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}}$ inserts $(V_\circ, PK_\circ, M_\circ, sk'_{1,b_1}, \ldots, sk'_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$.

For the first event, if $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$, it means that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$. As a result, $\mathcal{S}^{\mathcal{E}}$ would return $V_\circ$ by case 1 (red box above), which means this bad event only occurs if there is collision on uniformly sample string in $\mathbf{\Sigma}_\circ$, which is bounded by $\frac{q^2}{|\mathbf{\Sigma}_\circ|}$.

For the second event, note that if $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$, it means that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$. Under the condition that no $(SK_\circ \neq SK'_\circ)$ such that $\mathbf{oGen}(SK_\circ) = \mathbf{oSign}(SK'_\circ) = PK_\circ$ (bounded by Rule 1), we have that if $(sk_{1,b_1}, \ldots, sk_{n,b_n},) \neq (sk'_{1,b_1}, \ldots, sk'_{n,b_n})$, then the response is returned case 1 (red-box above). Thus, this bad event would not happen except for a collision ($\leq \frac{q^2}{|\mathbf{\Sigma}_\circ|}$).

For the last event, under the condition of Rule 4 (there is only one $\boldsymbol{pk}$ for $PK_\circ$), it's trivial that this bad event only occurs when there is collision on $\mathcal{H}_{\text{OneWay}}(sk_{i,b_i})$ and $\mathcal{H}_{\text{OneWay}}(sk'_{i,b_i})$. Concretely, the bad event occurs only if there exists $i \in [1, n]$ such that 1) $\mathcal{H}_{\text{OneWay}}(sk_{i,b_i}) = \mathcal{H}_{\text{OneWay}}(sk'_{i,b_i}) = pk_{i,b_i}$; 2) $sk_{i,b_i} \neq sk'_{i,b_i}$, which is bounded by $\frac{q^2}{2^\lambda}$.

Rule 8. Similar to Rule 5, there are six bad cases that might break this rule,

- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n})$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots, b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_{\mathcal{E}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
  and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V'_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n)$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;

It is trivial to note that those bad events would never happen unless there is a collision on $(sk_{1,b_1}, \ldots, sk_{n,b_n})$, and moreover the tuple $(sk_{1,b_1}, \ldots, sk_{n,b_n})$ in the procedure $\mathcal{S}^{\mathcal{E}^{-1}}$ is uniformly sampled, thus those bad events are bounded by $\frac{q^2}{2^{2n\lambda}}$.

Rule 9. This rule holds trivially by definition.

Rule 10. This rule holds as long as the differentiator cannot outputs $(SK_\circ \neq SK'_\circ)$ such that $\mathbf{oGen}(SK_\circ) = \mathbf{oGen}(SK'_\circ)$.

Rule 11. For this rule, it suffices to prove that for any $V_\circ$,

$$\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1 \Longleftrightarrow \mathcal{S}^{\Pi_\circ.\text{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 1.$$

Sub-rule: $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1 \Longrightarrow \mathcal{S}^{\Pi_\circ.\textit{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 1$. Given a $V_\circ$ such that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{E}^{-1}}(V_\circ)$ as follows:

- Strategy 1. If $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},) \in \mathbb{T}_\mathcal{E}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$.
- Strategy 2. if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$.
- Strategy 3. Otherwise, following the rest strategy in our simulator.

For strategy 3, trivial to note that, under the condition $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$, the response of $\mathcal{S}^{\mathcal{E}^{-1}}(V_\circ) = (\widehat{sk}_{1,b_1^*}, \ldots, \widehat{sk}_{n,b_n^*})$ is properly assigned such that

$$\begin{aligned}
&b_1^* \cdots b_n^* = \mathcal{S}^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ);\\
&\widehat{\boldsymbol{pk}} = \left(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\right) = \mathcal{S}^{\mathcal{P}^{-1}}(PK_\circ);\\
&\mathcal{H}_{\text{OneWay}}(\widehat{sk}_{i,b_i^*}) = \widehat{pk}_{i,b_i^*}.
\end{aligned}$$

which straightforwardly referring to $\mathcal{S}^{\Pi_\circ.\text{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 1$. Moreover, we note that the table $\mathbb{T}_{\mathcal{E}^{-1}}$ only records the tuple such that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$, thus the sub-rule also holds in strategy 2. And now, it's rest to prove the rule holds in strategy 1. In fact, there are three cases that the tuple $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},) \in \mathbb{T}_\mathcal{E}$,

1. $\mathcal{S}^\mathcal{E}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $ctr < n$;
2. $\mathcal{S}^\mathcal{E}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $ctr = n$;
3. $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$.

In the first case, we have that $V_\circ \twoheadleftarrow \mathbf{\Sigma}_\circ$, which means that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$ except for negligible probability ($\le \frac{q|\mathcal{SK}_\circ|}{\mathbf{\Sigma}_\circ}$). For the second one, we have that $ctr = n$, referring to $\mathcal{S}^{\Pi_\circ.\text{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 1$. For the last one, we have that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$. Thus this sub-rule holds except for negligible probability.

Sub-rule: $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1 \Longleftarrow \mathcal{S}^{\Pi_\circ.\textit{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 1$. Given an $V_\circ$ such that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{E}^{-1}}(V_\circ)$ as follows:

- Strategy 1. If $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},) \in \mathbb{T}_\mathcal{E}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$.
- Strategy 2. if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n},)$.
- Strategy 3. Otherwise, following the rest strategy in our simulator.

Immediately observe that this sub-rule holds in strategy 2, as $\mathbb{T}_{\mathcal{E}^{-1}}$ only records tuples such that $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$. For strategy 3, we note that if $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$, then the response of $\mathcal{S}^{\mathcal{E}^{-1}}(V_\circ) = (\widehat{sk}_{1,b_1^*}, \ldots, \widehat{sk}_{n,b_n^*})$ is uniformly sampled, which means $\mathcal{S}^{\Pi_\circ.\text{oVERIFY}}(PK_\circ, M_\circ, V_\circ) = 0$ except for negligible probability ($\le \frac{1}{2^t}$).

Next we show that this sub rule also holds for strategy 1. Concretely, there are three sub cases that the tuple $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},) \in \mathbb{T}_\mathcal{E}$,

1. Sub-case 1. $\mathcal{S}^\mathcal{E}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $ctr < n$;
2. Sub-case 2. $\mathcal{S}^\mathcal{E}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $ctr = n$;
3. Sub-case 3. $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n},)$ into $\mathbb{T}_\mathcal{E}$ where $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 0$.

For the second sub case, $V_\circ \leftarrow \mathbf{oSign}(SK_\circ, M_\circ)$, which means $\mathbf{oVerify}(PK_\circ, M_\circ, V_\circ) = 1$. For the third sub case, as $(\widehat{sk}_{1,b_1^*}, \ldots, \widehat{sk}_{n,b_n^*}, ^*)$ is uniformly sampled, which means $\mathcal{S}^{\Pi_\circ.\text{oVerify}}(PK_\circ, M_\circ, V_\circ) = 0$ except for negligible probability. Now we prove the first sub case, which includes several events:

1. $\neq 0 \cdots 0$;
2. $\exists SK_\circ \neq SK'_\circ$ s.t. $\mathbf{oGen}(SK_\circ) = \mathbf{oGen}(SK'_\circ) = PK_\circ$;
3. $\nexists (SK_\circ, \boldsymbol{sk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$;
4. $\nexists (PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
5. $0 < ctr < n$.

It's trivial to note that, if $\neq 0 \cdots 0$ or $0 < ctr < n$, then $\mathcal{S}^{\Pi_\circ.\text{oVerify}}(PK_\circ, M_\circ, V_\circ) = 0$. The second event is trivially bounded by $\frac{q^2}{|\mathcal{PK}_\circ|}$. Moreover, for the fourth event, the differentiator has no knowledge of or $b_1 \cdots$, thus even it has $\boldsymbol{sk}$,

$$\Pr[\mathcal{S}^{\Pi_\circ.\text{oVerify}}(PK_\circ, M_\circ, V_\circ) = 1] \leq \frac{q}{2^n}.$$

For the third event, the differentiator would note make a query $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK_\circ)$, thus the only way to obtain valid $\boldsymbol{sk}$ is the following:

- Step 1: $\mathcal{D}$ chooses M',
- Step 2: $\mathcal{D}$ makes queries $V'_\circ \leftarrow \mathbf{oSign}(SK_\circ, M')$,
- Step 3. $\mathcal{D}$ makes queries $(sk_{1,b'_1}, \ldots, sk_{n,b'_n})\mathcal{E}^{-1}(V'_\circ)$.

Note that, $\mathcal{D}$ can obtain only half of $\boldsymbol{sk}$, as $\mathbf{oSign}$ is one-time. In fact, for $SK_\circ$, $\mathcal{D}$ can only one signing query via the honest interface, thus unless $b_1 \cdots b_n = b'_1 \cdots b'_n$ ($\leq \frac{q^2}{2^n}$), there is at least one sk_{i,b_i} that is independent of $\mathcal{D}$'s view.

Combing together, we have that this sub-rule holds except for negligible probability.

5 Ideal Signature from Ideal One-time Signature

5.1 Construction

High-level ideas. We here present the construction of our digital signature scheme which is indifferentiable from the ideal signature (as defined in Figure 5). The details of our construction can be found on page 24, and next we illustrate some key design ideas.

The starting point here is the tree-based signature scheme. Similar to the argument in our one-time signature construction in previous section, to make the signature scheme indifferentiable from the ideal signature, we must eliminate the unnecessary structures in the tree-based signature scheme, since the adversary could start structure-abusing attacks. Our defending strategy is again to use ideal primitives including ideal one-time signature, random oracle, and ideal cipher, to properly isolate the state (including inputs, outputs, and internal state) in the key generation, signing, and verification algorithms of the tree-based signature scheme.

The tree-based signature scheme (Gen, Sign, Verify), uses one-time signature scheme (oGen, oSign, oVerify), and pseudorandom function $F_r(\cdot)$, as building blocks. We consider a full binary tree of depth n in the construction; and the basic idea in the tree-based construction is to use the verification- and signing-key (of a one-time signature scheme) to authenticate/sign two fresh instances (of the one-time signature scheme), and then use each of the instances to authenticate/sign two fresh instances, and so on. In this way, a binary tree of fresh instances of the one-time signature will be formed, in which each internal node authenticates its two children. The leaves of this tree will be used to sign actual messages, and here each leaf is used at most once. To sign a message, the resulting signature consists of (1) a one-time signature to the message which is authenticated with respect to the verification-key of a leaf, and (2) an authenticated path from the root to this leaf, i.e., a sequence of one-time verification-keys of all nodes in the path, in which each such verification-key is authenticated with respect to the verification-key of its parent. Concretely, in the key-generation algorithm GEN, the algorithm randomly samples sk and r, and computes $pk \leftarrow \text{oGEN}(sk)$; The signing-key is (r, sk) and the verification-key is pk. In the signing algorithm SIGN, on input the signing-key (r, sk), verification-key pk, and a message m, the algorithm first computes a path in the tree as $b_1 \cdots b_n \leftarrow F_r(\text{"path"}, pk||m)$, where $b_i \in \{0,1\}$, for $1 \leq i \leq n$. Second, the algorithm generates a signing-verification key-pair for each node of the path. That is, the algorithm computes $sk_{b_1 \cdots b_i \tau} \leftarrow F_r(\text{"node"}, b_1 \cdots b_i \tau)$ and $pk_{b_1 \cdots b_i \tau} \leftarrow \text{oGEN}(sk_{b_1 \cdots b_i \tau})$, for all $i = 0, \ldots, n-1$, and $\tau = 0, 1$. Third, the algorithm signs the path from the root to the leaf, i.e., computes $v_0 \leftarrow \text{oSIGN}(pk, sk, pk_0||pk_1)$, and computes $v_j \leftarrow \text{oSIGN}(pk_{b_1 \cdots b_j}, sk_{b_1 \cdots b_j}, pk_{b_1 \cdots b_j 0}||pk_{b_1 \cdots b_j 1})$, for all $1 \leq j \leq n-1$. Fourth, the algorithm sign the message to the leaf, i.e., computes $v \leftarrow \text{oSIGN}(pk_{b_1 \cdots b_n}, sk_{b_1 \cdots b_n}, m)$. The signature for message m is $\sigma = v||\langle v_0, pk_0, pk_1 \rangle|| \cdots ||\langle v_{n-1}, pk_{b_1 \cdots b_{n-1} 0}, pk_{b_1 \cdots b_{n-1} 1} \rangle$. Finally, in the verification algorithm, on input, verification key pk, message m and signature $\sigma = v||\langle v_0, pk_0, pk_1 \rangle|| \cdots ||\langle v_{n-1}, pk_{b_1 \cdots b_{n-1} 0}, pk_{b_1 \cdots b_{n-1} 1} \rangle$, the algorithm computes the path $b_1 \cdots b_n \leftarrow F_r(\text{"path"}, pk||m)$, where $b_i \in \{0,1\}$, for $1 \leq i \leq n$. The message-signature pair is valid if and only if the following conditions hold:

1) $\text{oVERIFY}(pk, pk_0||pk_1, v_0) = 1$,
2) $\text{oVERIFY}(pk_{b_1 \cdots b_{k-1} b_k}, pk_{b_1 \cdots b_k 0}||pk_{b_1 \cdots b_k 1}, v_k) = 1$, for all $1 \leq kn - 1$, and
3) $\text{oVERIFY}(pk_{b_1 \cdots b_n}, m, v) = 1$.

In order to upgrade the tree-based signature, the underlying one-time signature (oGEN,oSIGN,oVERIFY) will be replaced by ideal one-time signature $\mathbf{OSIG} = (\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$. In addition, the PRF $F_r(\cdot)$ will be replaced by random oracles. Finally, as in the construction in previous section, the structures in the "interfaces" of the key generation, signing, and verification algorithms of tree-based scheme, should be destroyed too. We use the ideal cipher to wrap up the generated signatures of tree-based scheme. We remark that, we do not need the random permutation to wrap up the generated verification key pk of tree-based scheme, since now pk is generated by the key generation of the ideal one-time signature, and the structure of pk has already been eliminated. Next, we specify some parameters:

- $n \geq \omega(\log \lambda)$, $\log |\mathcal{SK}_\circ| = 8n\lambda$, $\log |\mathcal{PK}_\circ| = 2n\lambda$;
- $\log |\mathbf{\Sigma}_\circ| = 2(\log |\mathcal{PK}_\circ| + \log |\mathcal{SK}_\circ| + \log |\mathcal{M}_\circ|)$;
- $\log |\mathcal{SK}| = (2n+5) \log |\mathcal{SK}_\circ|$, $\log |\mathcal{PK}| = 2n + 5 \log |\mathcal{PK}_\circ|$;
- $\log |\mathbf{\Sigma}| = (2n+2) \log |\mathcal{PK}| + (n+1) \log |\mathbf{\Sigma}_\circ| + 2(\log |\mathcal{SK}| + \log |\mathcal{PK}| + \log |\mathcal{M}|)$.

Building Blocks. Our scheme consists of several building blocks:

- An ideal one time signature $\{\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify}\}$, associated with secret key space $\mathcal{SK}_\circ$, verification key space $\mathcal{PK}_\circ$, message space $\mathcal{M}_\circ$ and signature space $\mathbf{\Sigma}_\circ$[7].
- $\mathcal{H}_{\text{skRoot}} : \{0,1\}^* \to \mathcal{SK}_\circ$ is a random oracle whose codomain matches the secret key space $\mathcal{SK}_\circ$;
- $\mathcal{H}_{\text{seed}} : \{0,1\}^* \to \{0,1\}^n$ is a random oracle.
- $\mathcal{H}_{\text{path}} : \{0,1\}^* \to \{0,1\}^n$ is a random oracle.
- $\mathcal{H}_{\text{msgLeaf}} : \{0,1\}^* \to \mathcal{M}_\circ$ is a random oracle whose codomain matches the message space $\mathcal{M}_\circ$;
- $\mathcal{H}_{\text{skLeftNode}} : \{0,1\}^* \to \mathcal{SK}_\circ$ is a random oracle whose codomain matches the secret key space $\mathcal{SK}_\circ$;
- $\mathcal{H}_{\text{skRightNode}} : \{0,1\}^* \to \mathcal{SK}_\circ$ is a random oracle whose codomain matches the secret key space $\mathcal{SK}_\circ$;
- $\mathcal{H}_{\text{skLeftLeaf}} : \{0,1\}^* \to \mathcal{SK}_\circ$ is a random oracle whose codomain matches the secret key space $\mathcal{SK}_\circ$;

[7]Our ideal one-time signature is deterministic, so it has no nonce space.

- $\mathcal{H}_{\text{skRightLeaf}} : \{0,1\}^* \to \mathcal{SK}_\circ$ is a random oracle whose codomain matches the secret key space $\mathcal{SK}_\circ$;
- $\mathcal{E} : \{0,1\}^{\log|\mathcal{PK}|+\log|\mathcal{M}|} \times \{0,1\}^{\log|\mathbf{\Sigma}|} \to \{0,1\}^{\log|\mathbf{\Sigma}|}$ is an ideal cipher model, where $\{0,1\}^{\log|\mathcal{PK}|+\log|\mathcal{M}|}$ is its key space and $\mathcal{E}^{-1}$ is its inverse.

5.1.1 Construction.

Now we are ready to build the indifferentiable signature, denoted as $\Pi = (\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify})$, using the tree-based structure. Formally,

$\Pi = (\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify})$

$PK \leftarrow \Pi.\textsc{Gen}(SK)$:

On input, signing key SK, compute the verification key PK, as follows:

```
1. $sk \leftarrow \mathcal{H}_{\text{skRoot}}(SK)$; $pk \leftarrow \mathbf{oGen}(sk)$; $PK \leftarrow pk$; output $PK$.
```

$V \leftarrow \Pi.\textsc{Sign}(PK, SK, M, R)$:

On input, verification key PK, signing key SK, message M, randomness R, compute the signature V, as follows:

```
1. $sk \leftarrow \mathcal{H}_{\text{skRoot}}(SK)$; $pk \leftarrow \mathbf{oGen}(sk)$;
   If $PK = pk$
      then $seed \leftarrow \mathcal{H}_{\text{seed}}(SK, M, R)$; $m \leftarrow \mathcal{H}_{\text{msgLeaf}}(PK||M||seed)$;
           $b_1 \cdots b_n b_{n+1} \leftarrow \mathcal{H}_{\text{path}}(PK||M||seed)$,
           where $b_i \in \{0,1\}$, for $1 \le i \le n+1$;
2. generate the signing-verification key-pair for each node of the path
   $path \leftarrow \emptyset$;
   $sk_0 \leftarrow \mathcal{H}_{\text{skLeftNode}}(SK||path)$; $pk_0 \leftarrow \mathbf{oGen}(sk_0)$;
   $sk_1 \leftarrow \mathcal{H}_{\text{skRightNode}}(SK||path)$; $pk_1 \leftarrow \mathbf{oGen}(sk_1)$;
   for $i \in [1, n-1]$
      $path \leftarrow b_1 \cdots b_i$;
      $sk_{b_1\cdots b_i 0} \leftarrow \mathcal{H}_{\text{skLeftNode}}(SK||path)$; $pk_{b_1\cdots b_i 0} \leftarrow \mathbf{oGen}(sk_{b_1\cdots b_i 0})$;
      $sk_{b_1\cdots b_i 1} \leftarrow \mathcal{H}_{\text{skRightNode}}(SK||path)$; $pk_{b_1\cdots b_i 1} \leftarrow \mathbf{oGen}(sk_{b_1\cdots b_i 1})$;
3. generate the signing-verification key-pair for each leaf[a]
   $path \leftarrow b_1 \cdots b_n$;
   $sk_{b_1\cdots b_n 0} \leftarrow \mathcal{H}_{\text{skLeftLeaf}}(SK||M||seed||path)$; $pk_{b_1\cdots b_n 0} \leftarrow \mathbf{oGen}(sk_{b_1\cdots b_n 0})$;
   $sk_{b_1\cdots b_n 1} \leftarrow \mathcal{H}_{\text{skRightLeaf}}(SK||M||seed||path)$; $pk_{b_1\cdots b_n 1} \leftarrow \mathbf{oGen}(sk_{b_1\cdots b_n 1})$;
4. sign the path
   $v_0 \leftarrow \mathbf{oSign}(pk, sk, pk_0||pk_1)$;
   for $j \in [1, n]$
      $v_j \leftarrow \mathbf{oSign}(pk_{b_1\cdots b_j}, sk_{b_1\cdots b_j}, pk_{b_1\cdots b_j 0}||pk_{b_1\cdots b_j 1})$;
5. sign the leaf
   $v \leftarrow \mathbf{oSign}(pk_{b_1\cdots b_{n+1}}, sk_{b_1\cdots b_{n+1}}, m)$;
6. output the signature
   $V \leftarrow \mathcal{E}\Big(PK||M,\ v||\langle v_0, pk_0, pk_1\rangle|| \cdots ||\langle v_n, pk_{b_1\cdots b_n 0}, pk_{b_1\cdots b_n 1}\rangle||seed||0\cdots 0\Big)$.
```

$\phi \leftarrow \Pi.\textsc{Verify}(PK, M, V)$:

On input, verification key PK, message M and signature V, operate as follows:

```
1. "unpack" the signature
   $\Big(\widehat{v}||\langle \widehat{v}_0, \widehat{pk}_0, \widehat{pk}_1\rangle|| \cdots ||\langle \widehat{v}_n, \widehat{pk}_{\widehat{b}_1\cdots\widehat{b}_n 0}, \widehat{pk}_{\widehat{b}_1\cdots\widehat{b}_n 1}\rangle||\widehat{seed}||\widehat{pad}\Big) \leftarrow \mathcal{E}^{-1}(PK||M, V)$;
2. $\widehat{m} \leftarrow \mathcal{H}_{\text{msgLeaf}}(PK||M||\widehat{seed})$;
   $\widehat{b}_1 \cdots \widehat{b}_{n+1} \leftarrow \mathcal{H}_{\text{path}}(PK||M||\widehat{seed})$;
3. output $\phi \leftarrow 1$ if and only if the following conditions hold:
      • $\widehat{pad} = 0 \ldots 0$,
```

- $\mathbf{oVerify}(PK, \widehat{pk}_0 || \widehat{pk}_1, \widehat{v}_0) = 1$,
- for $k \in [1, n]$,
  $\mathbf{oVerify}(\widehat{pk}_{\widehat{b}_1 \cdots \widehat{b}_{k-1} \widehat{b}_k}, \widehat{pk}_{\widehat{b}_1 \cdots \widehat{b}_k 0} || \widehat{pk}_{\widehat{b}_1 \cdots \widehat{b}_k 1}, \widehat{v}_k) = 1$,
- $\mathbf{oVerify}(\widehat{pk}_{\widehat{b}_1 \cdots \widehat{b}_n \widehat{b}_{n+1}}, \widehat{m}, \widehat{v}) = 1$.

otherwise output $\phi \leftarrow 0$.

[a]The secret keys at the bottom layer are randomized.

5.2 Security statement

Theorem 5.1 $\Pi = (\Pi.\mathbf{Gen}, \Pi.\mathbf{Sign}, \Pi.\mathbf{Verify})$ *is indifferentiable from the ideal signature* $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$*, in the model for, random oracles* $\mathcal{H}_{\text{skRoot}}, \mathcal{H}_{\text{path}}, \mathcal{H}_{\text{seed}}, \mathcal{H}_{\text{skLeftLeaf}}, \mathcal{H}_{\text{skRightLeaf}}, \mathcal{H}_{\text{skLeftNode}}, \mathcal{H}_{\text{skRightNode}}, \mathcal{H}_{\text{msgLeaf}}$*, ideal cipher* $(\mathcal{E}, \mathcal{E}^{-1})$*, and ideal one-time signature* $(\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$*. More precisely, there exists a simulator* $\mathcal{S}$ *such that for all* q*-query differentiator* $\mathcal{D}$*, we have*

$$\mathrm{Adv}^{\mathsf{indiff}}_{\Pi,\mathcal{S},\mathcal{D}} \leq \frac{2q^2}{2^n} + \frac{2q^2}{2^\lambda}.$$

The simulator makes at most q^2 *queries to its oracles.*

5.3 The simulator

We here describe the simulator and provide high-level proof ideas. According to the definition of indifferentiability, we have that, in the real world, the differentiator $\mathcal{D}$ has three honest interfaces ($\Pi.\textsc{Gen}, \Pi.\textsc{Sign}, \Pi.\textsc{Verify}$) and eleven adversarial interfaces ($\mathcal{H}_{\text{skRoot}}, \mathcal{H}_{\text{path}}, \mathcal{H}_{\text{seed}}, \mathcal{H}_{\text{skLeftLeaf}}, \mathcal{H}_{\text{skRightLeaf}}, \mathcal{H}_{\text{skLeftNode}}, \mathcal{H}_{\text{skRightNode}}, \mathcal{H}_{\text{msgLeaf}}, \mathcal{E}, \mathcal{E}^{-1}, \mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify}$). Therefore, to complete the proof, we build an efficient simulator $\mathcal{S}$ in the ideal world, such that 1) $\mathcal{S}$ has access to the ideal signature ($\mathbf{Gen}$, $\mathbf{Sign}$, $\mathbf{Verify}$) via the adversarial interfaces; 2) $\mathcal{S}$ simulates those eleven adversarial interfaces properly. Concretely, in the ideal world, the differentiator $\mathcal{D}$ has three honest interfaces ($\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify}$) and eleven adversarial interfaces ($\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}, \mathcal{S}^{\mathcal{H}_{\text{path}}}, \mathcal{S}^{\mathcal{H}_{\text{seed}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}, \mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}, \mathcal{S}^{\mathcal{H}_{\text{msgLeaf}}}, \mathcal{S}^{\mathcal{E}}, \mathcal{S}^{\mathcal{E}^{-1}}, \mathcal{S}^{\mathbf{oGen}}, \mathcal{S}^{\mathbf{oSign}}, \mathcal{S}^{\mathbf{oVerify}}$), and we prove that for any differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real world is close to the view in the ideal world. In the following, we illustrate the description of our simulator and then we give the high-level intuition of our proof strategy.

Simulator with oracle access to ideal signature ($\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify}$)

The simulator with oracle access to ideal signature ($\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify}$) , will provide the following interfaces:

```
$\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK)$:
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then return $sk$;
if $\exists(\diamond, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ s.t. $PK = \mathbf{Gen}(SK)$, then return $sk$;
$PK \leftarrow \mathbf{Gen}(SK)$; $sk \twoheadleftarrow \mathcal{SK}_{\mathsf{o}}$; $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \{(SK, sk, PK)\}$; return $sk$.

$\mathcal{S}^{\mathbf{oGen}}(sk)$:
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then return $PK$;
if $\exists(\diamond, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then return $PK$;
$SK \twoheadleftarrow \mathcal{SK}$; $PK \leftarrow \mathbf{Gen}(SK)$; $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \{(SK, sk, PK)\}$; return $PK$.

$\mathcal{S}^{\mathcal{H}_{\text{seed}}}(SK, M, R)$:
if $\exists(SK, M, R, V, seed) \in \mathbb{T}_{\mathcal{H}_{\text{seed}}}$, then return $seed$;
$V \leftarrow \mathbf{Sign}(SK, M, R)$; $PK \leftarrow \mathbf{Gen}(SK)$;
if $(V, PK, M, v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1 \cdots b_n 0}, pk_{b_1 \cdots b_n 1}\rangle, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then $\mathbb{T}_{\mathcal{H}_{\text{seed}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{seed}}} \cup \{(SK, M, R, V, seed)\}$; return $seed$;
$seed \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\text{seed}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{seed}}} \cup \{(SK, M, R, V, seed)\}$; return $seed$.

$\mathcal{S}^{\mathcal{H}_{\text{path}}}(PK, M, seed)$:
```

```
if $\exists(PK, M, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{H}_{\text{path}}}$,
    then return $b_1, \ldots, b_{n+1}$;
if $\exists(V, PK, M, v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1 \cdots b_n 0}, pk_{b_1 \cdots b_n 1}\rangle, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1, \ldots, b_{n+1}$;
$b_1, \ldots, b_{n+1} \twoheadleftarrow \{0,1\}^{n+1}$; $\mathbb{T}_{\mathcal{H}_{\text{path}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{path}}} \cup \big\{(PK, M, seed, b_1, \ldots, b_{n+1})\big\}$; return $b_1, \ldots, b_{n+1}$.
```

$\underline{\mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}(SK||path)}$: //(here $path = \emptyset$ or an i-bit string $path = b_1 \cdots b_i$)

```
$PK \leftarrow \mathbf{Gen}(SK)$;
if $\exists(PK, SK', sk', PK', path) \in \mathbb{T}_{\mathcal{H}_{\text{skLeftNode}}}$, then return $sk'$;
$SK' \twoheadleftarrow \mathcal{SK}$; $sk' \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;
$\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \big\{(SK', sk', PK')\big\}$;
$\mathbb{T}_{\mathcal{H}_{\text{skLeftNode}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skLeftNode}}} \cup \big\{(PK, SK', sk', PK', path)\big\}$;
return $sk'$.
```

$\underline{\mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}(SK||M||seed||b_1 \cdots b_n)}$:

```
$PK \leftarrow \mathbf{Gen}(SK)$; if $\exists(PK, M, seed, SK', sk', PK', b_1, \ldots, b_n) \in \mathbb{T}_{\mathcal{H}_{\text{skLeftLeaf}}}$, then return $sk'$;
$SK' \twoheadleftarrow \mathcal{SK}$; $sk' \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$; $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \big\{(SK', sk', PK')\big\}$;
$\mathbb{T}_{\mathcal{H}_{\text{skLeftLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skLeftLeaf}}} \cup \big\{(PK, M, seed, SK', sk', PK', b_1, \ldots, b_n)\big\}$; return $sk'$.
```

$\underline{\mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}(SK||path)}$: //(here $path = \emptyset$ or an i-bit string $path = b_1 \cdots b_i$)

```
$PK \leftarrow \mathbf{Gen}(SK)$,
if $\exists(PK, SK', sk', PK', path) \in \mathbb{T}_{\mathcal{H}_{\text{skRightNode}}}$,
    then return $sk'$;
$SK' \twoheadleftarrow \mathcal{SK}$; $sk' \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;
$\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \big\{(SK', sk', PK')\big\}$;
$\mathbb{T}_{\mathcal{H}_{\text{skRightNode}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRightNode}}} \cup \big\{(PK, SK', sk', PK', path)\big\}$; return $sk'$.
```

$\underline{\mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}(SK||M||seed||b_1 \cdots b_n)}$:

```
$PK \leftarrow \mathbf{Gen}(SK)$;
if $\exists(PK, M, seed, SK', sk', PK', b_1, \ldots, b_n) \in \mathbb{T}_{\mathcal{H}_{\text{skRightLeaf}}}$, then return $sk'$;
$SK' \twoheadleftarrow \mathcal{SK}$; $sk' \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$; $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \big\{(SK', sk', PK')\big\}$;
$\mathbb{T}_{\mathcal{H}_{\text{skRightLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRightLeaf}}} \cup \big\{(PK, M, seed, SK', sk', PK', b_1, \ldots, b_n)\big\}$; return $sk'$.
```

$\underline{\mathcal{S}^{\mathcal{H}_{\text{msgLeaf}}}(PK||M||seed)}$:

```
if $\exists(PK, M, seed, m) \in \mathbb{T}_{\mathcal{H}_{\text{msgLeaf}}}$, then return $m$;
$m \twoheadleftarrow \mathcal{M}_{\mathbf{o}}$; $\mathbb{T}_{\mathcal{H}_{\text{msgLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{msgLeaf}}} \cup \big\{(PK, M, seed, m)\big\}$; return $m$.
```

$\underline{\mathcal{S}^{\mathbf{oSign}}(sk, \alpha)}$: //(here α can be a message i.e., $\alpha = m$, or two verification keys, i.e., $\alpha = pk_0^*||pk_1^*$)

```
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$,
    then $pk \leftarrow PK$;     else $SK \twoheadleftarrow \mathcal{SK}$; $PK \leftarrow \mathbf{Gen}(SK)$; $pk \leftarrow PK$; $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{skRoot}}} \cup \big\{(SK, sk, PK)\big\}$;
if $\exists(pk, sk, \alpha, v) \in \mathbb{T}_{\mathbf{oSign}}$,
    then return $v$;
$v \twoheadleftarrow \mathbf{\Sigma}_{\mathbf{o}}$; $\mathbb{T}_{\mathbf{oSign}} \leftarrow \mathbb{T}_{\mathbf{oSign}} \cup \big\{(pk, sk, \alpha, v)\big\}$; return $v$.
```

$\underline{\mathcal{S}^{\mathbf{oVerify}}(pk, \alpha, v)}$ //(here α can be a message i.e., $\alpha = m$, or two verification keys, i.e., $\alpha = pk_0^*||pk_1^*$)

```
if $\exists(pk, sk, \alpha, v) \in \mathbb{T}_{\mathbf{oSign}}$,
    then return 1;
    else, return 0.
```

$\underline{\mathcal{S}^{\mathcal{E}}(PK||M, v||\langle v_0, pk_0, pk_1\rangle|| \cdots ||\langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1}\rangle||seed||pad)}$:

```
if $\exists(V, PK, M, v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1}\rangle, seed, pad) \in \mathbb{T}_{\mathcal{E}}$, then return $V$.
if $\exists(V, PK, M, v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1}\rangle, seed, pad, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $V$.
$ctr_1 \leftarrow 0$; $ctr_2 \leftarrow 0$;
if $pad \neq 0, \ldots, 0$,
    then goto Case 1;
```

```
if $\exists (SK \neq SK')$ s.t. $\big((SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}\big) \wedge \big((SK', sk', PK) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}\big)$
    then goto Case 1;
if $\nexists (SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}$,
    then goto Case 1;
    else $SK^* \leftarrow SK$; $sk^* \leftarrow sk$;
if $\nexists (PK, M, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{H}_{\mathrm{path}}}$,
    then $(b_1, \ldots, b_{n+1}) \leftarrow \{0,1\}^{n+1}$; $\mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \cup \{(PK, M, seed, b_1, \ldots, b_{n+1})\}$, $b_1^*, \ldots, b_{n+1}^* \leftarrow b_1, \ldots, b_{n+1}$;
    else $b_1^*, \ldots, b_{n+1}^* \leftarrow b_1, \ldots, b_{n+1}$;
if $\nexists (PK, M, seed, m) \in \mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}}$,
    then goto Case 1;
    else $m^* \leftarrow m$;
//$\mathcal{S}^{\mathbf{oVerify}}$ only checks the table $\mathbb{T}_{\mathbf{oSign}}$
if $\mathcal{S}^{\mathbf{oVerify}}(pk_{b_1^* \cdots b_n^* b_{n+1}^*}, m^*, v) = 1$, then $ctr_1 \leftarrow ctr_1 + 1$;
if $\mathcal{S}^{\mathbf{oVerify}}(PK, pk_0 || pk_1, v_0) = 1$, then $ctr_1 \leftarrow ctr_1 + 1$;
for $i = 1$ to $n$,
    if $\mathcal{S}^{\mathbf{oVerify}}(pk_{b_1^* \cdots b_i^*}, pk_{b_1^* \cdots b_i^* 0} || pk_{b_1^* \cdots b_i^* 1}, v_i) = 1$, then $ctr_1 \leftarrow ctr_1 + 1$;

//check the validity of every signature in the path
if $\exists (SK, M, R, seed) \in \mathbb{T}_{\mathcal{H}_{\mathrm{seed}}}$ s.t. $\mathbf{Gen}(SK) = PK$,
    then $R^* \leftarrow R$; $ctr_2 \leftarrow ctr_2 + 1$;
for $i = 0$ to $n - 1$,
    if $\exists (PK, SK', sk', pk_{b_1^* \cdots b_i^* 0}, b_1^*, \ldots, b_i^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}}$, then $ctr_2 \leftarrow ctr_2 + 1$;
    if $\exists (PK, SK', sk', pk_{b_1^* \cdots b_i^* 1}, b_1^*, \ldots, b_i^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}}$, then $ctr_2 \leftarrow ctr_2 + 1$;
if $\exists (PK, SK, M, seed, SK', sk', pk_{b_1^* \cdots b_n^* 0}, b_1^*, \ldots, b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}}$, then $ctr_2 \leftarrow ctr_2 + 1$;
if $\exists (PK, SK, M, seed, SK', sk', pk_{b_1^* \cdots b_n^* 1}, b_1^*, \ldots, b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}}$, then $ctr_2 \leftarrow ctr_2 + 1$;

//check the validity of every verification key in the path
if $ctr_1 < n + 2$,
    then goto Case 1;
    else if $ctr_1 = n + 2 \wedge ctr_2 < 2n + 3$,
        then goto Case 2;
        else goto Case 3;

Case 1: $V \twoheadleftarrow \mathbf{\Sigma}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \big\{(V, PK, M, v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad)\big\}$; return $V$.

Case 2: $R^* \twoheadleftarrow \mathcal{R}$; $V \leftarrow \mathbf{Sign}(SK^*, M, R^*)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \big\{(V, PK, M, v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad)\big\}$; return $V$.

Case 3: $V \leftarrow \mathbf{Sign}(SK^*, M, R^*)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \big\{(V, PK, M, v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad)\big\}$; return $V$.
```

```
$\mathcal{S}^{\mathcal{E}^{-1}}(PK || M, V)$:
if $\exists (V, PK, M, v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
        then return $(v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad)$;
if $\exists (V, PK, M, v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1} \rangle, seed, pad)$;

if $\mathbf{Verify}(PK, M, V) = 0$, //respond with random strings for invalid signature
    then $v \twoheadleftarrow \mathbf{\Sigma_o}$; $v_i \twoheadleftarrow \mathbf{\Sigma_o}$ for all $i \in \{1, \ldots, n\}$; $pk_{\beta 0} \twoheadleftarrow \mathcal{PK}_{\mathbf{o}}$; $pk_{\beta 1} \twoheadleftarrow \mathcal{PK}_{\mathbf{o}}$;
        $seed \twoheadleftarrow \{0,1\}^n$; $pad \twoheadleftarrow \{0,1\}^t$; $b_1, \ldots, b_{n+1} \twoheadleftarrow \{0,1\}^{n+1}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \big\{(v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1 \cdots b_n 0}, pk_{b_1 \cdots b_n 1} \rangle, seed, pad)\big\}$;
        return $(v, \langle v_0, pk_0, pk_1 \rangle, \ldots, \langle v_n, pk_{b_1 \cdots b_n 0}, pk_{b_1 \cdots b_n 1} \rangle, seed, pad)$.
If $\exists (SK, M, R, seed) \in \mathbb{T}_{\mathcal{H}_{\mathrm{seed}}}$ s.t. $\mathbf{Sign}(SK, M, R) = V$,
    then $seed^* \leftarrow seed$;
    else $seed^* \twoheadleftarrow \{0,1\}^n$;
```

```
If $\big(\exists(SK, M, R, seed) \in \mathbb{T}_{\mathcal{H}_{\mathrm{seed}}} \wedge (PK, M, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{H}_{\mathrm{path}}}$ s.t. $\mathbf{Sign}(SK, M, R) = V\big)$ //compute the path in the tree
   then $b_1^*, \ldots, b_{n+1}^* \leftarrow b_1, \ldots, b_{n+1}$; $\mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \cup \{(PK, M, seed^*, b_1^*, \ldots, b_{n+1}^*)\}$;
   else $b_1^*, \ldots, b_{n+1}^* \twoheadleftarrow \{0,1\}^{n+1}$; $\mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{path}}} \cup \{(PK, M, seed^*, b_1^*, \ldots, b_{n+1}^*)\}$.
If $\big(\exists(SK, M, R, seed) \in \mathbb{T}_{\mathcal{H}_{\mathrm{seed}}} \wedge (PK, M, seed, m) \in \mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}}$ s.t. $\mathbf{Sign}(SK, M, R) = V\big)$, //compute the message in the leaf
   then $m^* \leftarrow m$; $\mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}} \cup \{(PK, M, seed^*, m^*)\}$;
   else $m^* \twoheadleftarrow \mathcal{M}_{\mathbf{o}}$; $\mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{msgLeaf}}} \cup \{(PK, M, seed^*, m^*)\}$;

if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}$,
   then $sk^* \leftarrow sk$;
   else $sk \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $\mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \cup \{(\diamond, sk, PK)\}$; $sk^* \leftarrow sk$;

for $i = 0$ to $n-1$,
   if $\exists(PK, SK', sk', PK', b_1^*, \ldots, b_i^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}}$,
      then $sk_{b_1^* \cdots b_i^* 0} \leftarrow sk'$; $pk_{b_1^* \cdots b_i^* 0} \leftarrow PK'$;
      else $SK' \twoheadleftarrow \mathcal{SK}$; $sk \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;      $\mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \cup \{(SK', sk', PK')\}$;
         $\mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}} \cup \{(PK, SK', sk', PK', b_1^*, \ldots, b_i^*)\}$;
         $sk_{b_1^* \cdots b_i^* 0} \leftarrow sk'$; $pk_{b_1^* \cdots b_i^* 0} \leftarrow PK'$; //compute secret/public keys of the nodes in the path
   if $\exists(PK, SK', sk', PK', b_1^*, \ldots, b_i^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}}$,
      then $sk_{b_1^* \cdots b_i^* 1} \leftarrow sk'$; $pk_{b_1^* \cdots b_i^* 1} \leftarrow PK'$;
      else $SK' \twoheadleftarrow \mathcal{SK}$; $sk \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;
         $\mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \cup \{(SK', sk', PK')\}$;
         $\mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}} \cup \{(PK, SK', sk', PK', b_1^*, \ldots, b_i^*)\}$;
         $sk_{b_1^* \cdots b_i^* 1} \leftarrow sk'$; $pk_{b_1^* \cdots b_i^* 1} \leftarrow PK'$; //compute secret/public keys of the nodes in the path

if $\big(\exists(PK, M, seed^*, SK', sk', PK', b_1^*, \ldots, b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}}\big)$,   then $sk_{b_1^* \cdots b_n^* 0} \leftarrow sk'$; $pk_{b_1^* \cdots b_n^* 0} \leftarrow PK'$;
   else $SK' \twoheadleftarrow \mathcal{SK}$; $sk \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;
      $\mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \cup \{(SK', sk', PK')\}$;
      $\mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}} \cup \{(PK, M, seed^*, SK', sk', PK', b_1^*, \ldots, b_n^*)\}$;
      $sk_{b_1^* \cdots b_n^* 0} \leftarrow sk'$; $pk_{b_1^* \cdots b_n^* 0} \leftarrow PK'$; //compute secret/public keys of the leaves in the path
if $\exists(PK, M, seed^*, SK', sk', PK', b_1^*, \ldots, b_i^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}}$,
      then $sk_{b_1^* \cdots b_n^* 1} \leftarrow sk'$; $pk_{b_1^* \cdots b_n^* 1} \leftarrow PK'$;
      else $SK' \twoheadleftarrow \mathcal{SK}$; $sk \twoheadleftarrow \mathcal{SK}_{\mathbf{o}}$; $PK' \leftarrow \mathbf{Gen}(SK')$;
         $\mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}} \cup \{(SK', sk', PK')\}$;
         $\mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}} \cup \{(PK, M, seed^*, SK', sk', PK', b_1^*, \ldots, b_n^*)\}$;
         $sk_{b_1^* \cdots b_n^* 1} \leftarrow sk'$; $pk_{b_1^* \cdots b_n^* 1} \leftarrow PK'$; //compute secret/public keys of the leaves in the path
//compute every signature in the path
if $\exists(sk^*, pk_0||pk_1, v_0) \in \mathbb{T}_{\mathbf{oSign}}$,
   then $v_0^* \leftarrow v_0$;
   else $v_0 \twoheadleftarrow \mathbf{\Sigma_o}$; $v_0^* \leftarrow v_0$; $\mathbb{T}_{\mathbf{oSign}} \leftarrow \mathbb{T}_{\mathbf{oSign}} \cup \{(sk^*, pk_0||pk_1, v_0)\}$;
for $i = 1$ to $n$,
   if $\exists(sk_{b_1^* \cdots b_i^*}, pk_{b_1^* \cdots b_i^* 0}||pk_{b_1^* \cdots b_i^* 1}, v_i) \in \mathbb{T}_{\mathbf{oSign}}$,
      then $v_i^* \leftarrow v_i$;
      else $v_i \twoheadleftarrow \mathbf{\Sigma_o}$; $v_i^* \leftarrow v_i$; $\mathbb{T}_{\mathbf{oSign}} \leftarrow \mathbb{T}_{\mathbf{oSign}} \cup \{(sk_{b_1^* \cdots b_i^*}, pk_{b_1^* \cdots b_i^* 0}||pk_{b_1^* \cdots b_i^* 1}, v_i)\}$;
if $\exists(sk_{b_1^* \cdots b_{n+1}^*}, m, v) \in \mathbb{T}_{\mathbf{oSign}}$,
   then $v^* \leftarrow v$;
   else $v \twoheadleftarrow \mathbf{\Sigma_o}$; $v^* \leftarrow v$; $\mathbb{T}_{\mathbf{oSign}} \leftarrow \mathbb{T}_{\mathbf{oSign}} \cup \{(sk_{b_1^* \cdots b_{n+1}^*}, m, v)\}$;
$pad^* \leftarrow 0, \ldots, 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V, PK, M, v^*, \langle v_0^*, pk_0, pk_1\rangle, \ldots, \langle v_n^*, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1}\rangle, seed^*, pad^*, b_1^*, \ldots, b_{n+1}^*)\}$;
return $(v^*, \langle v_0^*, pk_0, pk_1\rangle, \ldots, \langle v_n^*, pk_{b_1^* \cdots b_n^* 0}, pk_{b_1^* \cdots b_n^* 1}\rangle, seed^*, pad^*)$.
```

We immediately note that, if the differentiator $\mathcal{D}$ makes q queries via the adversarial interfaces, then our simulator $\mathcal{S}$ makes at most q queries to **Gen**, **Sign**, **Verify**. The simulator $\mathcal{S}$ only keeps nine tables with size at most $2nq$ ($\mathbb{T}_{\mathcal{H}_{\mathrm{skLeftNode}}}, \mathbb{T}_{\mathcal{H}_{\mathrm{skRightNode}}}, \mathbb{T}_{\mathcal{H}_{\mathrm{skLeftLeaf}}}, \mathbb{T}_{\mathcal{H}_{\mathrm{skRightLeaf}}}$), which means that the simulator $\mathcal{S}$ is efficient. In the following, we present the intuitive ideas to argue that why $\mathcal{S}$ works. Note that, in the real-world game, $\mathcal{H}_{\mathrm{skRoot}}$, $\mathcal{H}_{\mathrm{seed}}$, $\mathcal{H}_{\mathrm{path}}$, $\mathcal{H}_{\mathrm{msgLeaf}}$, $\mathcal{H}_{\mathrm{skLeftNode}}$, $\mathcal{H}_{\mathrm{skLeftLeaf}}$, $\mathcal{H}_{\mathrm{skRightNode}}$, and $\mathcal{H}_{\mathrm{skRightLeaf}}$ are random oracles, $\mathcal{E}$ is an ideal cipher model associated with its inverse $\mathcal{E}^{-1}$, and $(\mathbf{oGen}, \mathbf{oSign}, \mathbf{oVerify})$ is an ideal one-time signature. Hence, the responses of a proper simulator should follow the following rules:

1. The responses of the simulated interfaces, $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}, \mathcal{S}^{\mathcal{H}_{\text{seed}}}, \mathcal{S}^{\mathcal{H}_{\text{path}}}, \mathcal{S}^{\mathcal{H}_{\text{msgLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}$, and $\mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}$ are statistically close to the uniform;
2. The responses of $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}, \mathcal{S}^{\mathcal{H}_{\text{seed}}}, \mathcal{S}^{\mathcal{H}_{\text{path}}}, \mathcal{S}^{\mathcal{H}_{\text{msgLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}$, $\mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}, and \mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}$ are consistent;
3. For fixed $(PK||M)$, the responses of $(\mathcal{S}^{\mathcal{E}}, \mathcal{S}^{\mathcal{E}^{-1}})$ are statistically close to those of an ideal cipher;
4. There exists no $(v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1\cdots b_n 0}, pk_{b_1\cdots b_n 1}\rangle, seed, pad) \neq (v', \langle v'_0, pk'_0, pk'_1\rangle, \ldots, \langle v'_n, pk'_{b'_1\cdots b'_n 0}, pk'_{b'_1\cdots b'_n 1}\rangle, seed', pad')$ such that
$$\begin{aligned}&\mathcal{S}^{\mathcal{E}}\Big(PK||M,\ v||\langle v_0, pk_0, pk_1\rangle||\cdots||\langle v_n, pk_{b_1\cdots b_n 0}, pk_{b_1\cdots b_n 1}\rangle||seed||pad\Big)\\ &= \mathcal{S}^{\mathcal{E}}\Big(PK||M,\ v'||\langle v'_0, pk'_0, pk'_1\rangle||\cdots||\langle v'_n, pk'_{b'_1\cdots b'_n 0}, pk'_{b'_1\cdots b'_n 1}\rangle||seed'||pad\Big).\end{aligned}$$
5. There exists no $V \neq V'$ such that
$$\mathcal{S}^{\mathcal{E}^{-1}}(PK||M, V) = \mathcal{S}^{\mathcal{E}^{-1}}(PK||M, V').$$
6. The responses of $(\mathcal{S}^{\mathbf{oGen}}, \mathcal{S}^{\mathbf{oSign}}, \mathcal{S}^{\mathbf{oVerify}})$ are statistically close to an ideal one-time signature.
7. $\mathbf{Gen}(SK) = \mathcal{S}^{\Pi.\text{GEN}}(SK)$.
8. $\mathbf{Sign}(SK, M, R) = \mathcal{S}^{\Pi.\text{SIGN}}(SK, M, R)$.
9. $\mathbf{Verify}(PK, M, V) = \mathcal{S}^{\Pi.\text{VERIFY}}(PK, M, V)$.

Next, we illustrate why and how $\mathcal{S}$ achieves these eleven rules.

Rule 1. We here show that the responses of $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}, \mathcal{S}^{\mathcal{H}_{\text{seed}}}, \mathcal{S}^{\mathcal{H}_{\text{path}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}$, $\mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}, \mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}$, and $\mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}$ are well distributed.

The response of $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}$. By definition, $\mathcal{S}$ responds to $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK)$, by either using the table $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ or returning a uniformly sampled sk. Concretely,

- Strategy 1. If there is a tuple $(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then $\mathcal{S}$ responds with sk;
- Strategy 2. If there is a tuple $(\diamond, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ such that $\mathbf{Gen}(SK) = PK$, then $\mathcal{S}$ responds with sk;
- Strategy 3. $\mathcal{S}$ responds with a uniformly sampled sk.

It is trivial to note that, the response of the last case is well-distributed. Next, we analyze the first two cases. For the first case, we note that the tuple (SK, sk, PK) is inserted into $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ by two procedures, $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}$ and $\mathcal{S}^{\mathbf{oSign}}$: if the tuple is inserted by $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}$, then the response is fine as sk is uniformly sampled, while if the tuple is inserted by $\mathcal{S}^{\mathbf{oSign}}$, then the response might not be uniformly distributed, as sk can be chosen by the adversary. Fortunately, we observe that if the tuple (SK, sk, PK) is inserted by $\mathcal{S}^{\mathbf{oSign}}$, then SK is uniformly sampled by $\mathcal{S}$ and independent of the differentiator's view. Thus, the differentiator would not make such a query except for negligible probability ($\leq \frac{q}{|\mathcal{SK}_{\text{o}}|}$). For the second case, the tuple $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_{\text{o}})$ is inserted by $\mathcal{S}^{\mathcal{E}^{-1}}$, and sk is uniformly sampled in both procedures, which means that the response is well-distributed. However, if the differentiator has $(SK \neq SK')$ such that $\mathbf{Gen}(SK) = \mathbf{Gen}(SK') = PK$, then $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK_{\text{o}}) = \mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK'_{\text{o}})$, which induces a collision. While, due to the definition of ideal signature, this bad event would not occur except for negligible probability ($\leq \frac{q^2}{|\mathcal{PK}|}$).

The response of $\mathcal{S}^{\mathcal{H}_{\text{seed}}}$. By definition, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{H}_{\text{seed}}}$ by either using the table $\mathbb{T}_{\mathcal{H}_{\text{seed}}}, \mathbb{T}_{\mathcal{E}^{-1}}$ or returning a uniformly sampled $seed$. It is trivial to note that in all the cases, $seed$ is uniformly sampled, which means that the response is well-distributed.

Moreover, within the same analysis, it's apparent that the responses of $\mathcal{S}^{\mathcal{H}_{\text{path}}}$ are well distributed.

The response of $\mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}$. By definition, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}$ by either using the table $\mathbb{T}_{\mathcal{H}_{\text{skLeftNode}}}$ or returning a uniformly sampled $seed$. However, if the differentiator has $(SK \neq SK')$ such that $\mathbf{Gen}(SK) = \mathbf{Gen}(SK') = PK$, then for any $b_1, \ldots, b_{n+1}$, we have that

$$\mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}(SK||b_1, \ldots, b_{n+1}) = \mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK'||b_1, \ldots, b_{n+1}),$$

which breaks the rule immediately. In fact, this bad event would induce a collision on PK, and due to the definition of the ideal signature, it never occurs except for negligible probability ($\leq \frac{q^2}{|\mathcal{PK}|}$). Within the same analysis, the responses of $\mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}$, $\mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}$ and $\mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}$ are well-distributed.

Rule 2. Trivial to note that the responses of $\mathcal{S}^{\mathcal{H}_{\text{seed}}}, \mathcal{S}^{\mathcal{H}_{\text{path}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftNode}}}, \mathcal{S}^{\mathcal{H}_{\text{skLeftLeaf}}}$, $\mathcal{S}^{\mathcal{H}_{\text{skRightNode}}}$, and $\mathcal{S}^{\mathcal{H}_{\text{skRightLeaf}}}$ are consistent. And for $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}$, we observe that the the only bad case that causes inconsistency for $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}(SK)$ is that the table $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ records two tuples: (SK, sk, PK) and $(\diamond, sk, PK)$. Concretely, $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(\diamond, sk, PK)$ and then $\mathcal{S}^{\mathbf{oGen}}$ inserts (SK, sk', PK). Observe that, SK is uniformly sampled by $\mathcal{S}$, which means that collision on PK would not happen except for negligible probability ($\leq \frac{q^2}{|PK|}$).

Rule 3. Same as the analysis in Sec 3, it suffices to prove that, for any fixed $(PK||M)$, any pair $\Big((v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1\cdots b_n 0}, pk_{b_1\cdots b_n 1}\rangle, seed, pad),\ V_\circ\Big)$[8] such that $\mathcal{S}^{\mathcal{E}}(PK||M, \boldsymbol{v}) = V$ either $\boldsymbol{v}$ or V are uniformly distributed. There are two cases:

1. $(V, PK, M, \boldsymbol{v}) \in \mathbb{T}_{\mathcal{E}}$.
2. $(V, PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$

For the former case where the tuple is in $\mathbb{T}_{\mathcal{E}}$, we note that if the tuple is inserted by $\mathcal{S}^{\mathcal{E}}$, then there are three sub-cases: 1) V is either random sample (red box); 2) $V = \mathbf{Sign}(SK, M, R^*)$ where $R^* \twoheadleftarrow \mathcal{R}$; 3) $V = \mathbf{Sign}(SK, M, R^*)$ for some R^* recorded in the table. Trivial to note that, in all of the sub cases, V is uniformly distributed. And if the tuple is inserted by $\mathcal{S}^{\mathcal{E}^{-1}}$, then by definition, we have that $\boldsymbol{v}$ is uniformly sampled.

For the latter case, $\boldsymbol{v}$ of course is not uniformly distributed as $pad = 0\cdots 0$, while we have that, for each tuple in $\mathbb{T}_{\mathcal{E}^{-1}}$, $\mathbf{Verify}(PK||M, V) = 1$, which refers to that $V = \mathbf{Sign}(SK, M, R)$ where $\mathbf{Gen}(SK) = PK$. Thus V is well-distributed.

Rule 4. There are three bad events that might break the rule,

- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$, then $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK, M, \boldsymbol{v}', b'_1 \cdots b'_{n+1})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$, then $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK, M, \boldsymbol{v}')$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$, and then$\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK, M, \boldsymbol{v}')$ into $\mathbb{T}_{\mathcal{E}}$.

For the first event, note that if $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$, it means that $\mathbf{Verify}(PK, M, V) = 0$. As a result, $\mathcal{S}^{\mathcal{E}}$ would return V only by case 1 in red box (if V is returned by yellow or green box, then $\mathbf{Verify}(PK, M, V) = 1$), which means this bad event only occurs if there is collision on uniformly sample string in $\boldsymbol{\Sigma}$, which is bounded by $\frac{q^2}{|\boldsymbol{\Sigma}|}$.

[8]Below we will denote $\boldsymbol{v}$ as $(v, \langle v_0, pk_0, pk_1\rangle, \ldots, \langle v_n, pk_{b_1\cdots b_n 0}, pk_{b_1\cdots b_n 1}\rangle, seed, pad)$

For the second event, note that if $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V, PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$, it means that $\mathbf{Verify}(PK, M, V) = 1$. Under the condition that no $(SK \neq SK')$ such that $\mathbf{Gen}(SK) = \mathbf{Sign}(SK') = PK$ (bounded by Rule 1), we have that if $(\boldsymbol{v}) \neq (\boldsymbol{v}')$, then the response is returned by case 2 in yellow box (if V is returned by case 3 in green box, then $\boldsymbol{v} = \boldsymbol{v}'$). And in case 2, the randomness R is uniformly sampled, hence this bad event would not happen except for a collision ($\leq \frac{q^2}{|\mathcal{R}|}$).

For the last event, under the condition that of rule 6 (the responses of $(\mathcal{S}^{\mathbf{oGen}}, \mathcal{S}^{\mathbf{oSign}}, \mathcal{S}^{\mathbf{oVerify}})$ are statistically close to an ideal one-time signature), this bad event is trivially bounded by a collision on the signature.

Rule 5. Similar to Rule 5, there are six bad cases that might break this rule,

- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
- $\mathcal{S}^{\mathcal{E}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;
- $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(V, PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$ and then $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V', PK, M, \boldsymbol{v}, b_1, \ldots, b_{n+1})$ into $\mathbb{T}_{\mathcal{E}^{-1}}$;

It is trivial to note that those bad events would never happen unless there is a collision on $\boldsymbol{v}$, and moreover the tuple $\boldsymbol{v})$ in the procedure $\mathcal{S}^{\mathcal{E}^{-1}}$ is uniformly sampled, thus those bad events are bounded by $\frac{q^2}{2^{2n\lambda}}$.

Rule 6. For this rule, we need to prove the following:

1. *Rule 6.1:* The responses of $\mathcal{S}^{\mathbf{oGen}}, \mathcal{S}^{\mathbf{oSign}}, \mathcal{S}^{\mathbf{oVerify}}$ are consistent;
2. *Rule 6.2:* The responses of $\mathcal{S}^{\mathbf{oGen}}$ and $\mathcal{S}^{\mathbf{oSign}}$ are close to uniform.
3. *Rule 6.3:* For any SK and M,
$$\mathcal{S}^{\mathbf{oVerify}}(\mathcal{S}^{\mathbf{oGen}}(SK), M, \mathcal{S}^{\mathbf{oSign}}(SK, M)) = 1$$
4. *Rule 6.4:* For any PK, M, V, if $\nexists SK$ such that $\mathcal{S}^{\mathbf{oGen}}(SK) = PK \wedge V = \mathcal{S}^{\mathbf{oSign}}(SK, M)$, then
$$\mathcal{S}^{\mathbf{oVerify}}(PK, M, V) = 0$$

Next, we present our analysis one by one.
Rule 6.1. Trivial to note that the responses of $\mathcal{S}^{\mathbf{oSign}}$ and $\mathcal{S}^{\mathbf{oVerify}}$ are consistent. And for $\mathcal{S}^{\mathbf{oGen}}$, the only bad case that break the rule is that: $\mathcal{S}^{\mathcal{E}^{-1}}$ first inserts $(\diamond, sk, PK)$ into $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$ and then $\mathcal{S}^{\mathcal{H}_{\text{skRoot}}}$ inserts (SK', sk, PK') into $\mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$. Note that sk in the latter tuple is uniformly sampled, thus this bad event never occurs except for negligible probability ($\leq \frac{q}{|\mathcal{SK}_{\text{o}}|}$).

Rule 6.2. Trivial to note that the responses of $\mathcal{S}^{\mathbf{oSign}}$ are uniformly sampled. And for $\mathcal{S}^{\mathbf{oGen}}$, there are three strategies:

1. Strategy 1. If there is a tuple $(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then $\mathcal{S}$ responds with PK;
2. Strategy 2. If there is a tuple $(\diamond, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$, then $\mathcal{S}$ responds with PK;
3. Strategy 3. $\mathcal{S}$ responds with $PK \leftarrow \mathbf{Gen}(SK)$.

Easy to observe that in both strategy 1 and strategy 3, the response are uniform. While in strategy 2, PK could be chosen by the adversary, which means the response is not uniform. Fortunately, in such a stratety sk is uniform sampled by $\mathcal{S}$ and the differentiator would not make such a query except for negligible probability ($\leq \frac{q}{|\mathcal{SK}_\circ|}$).

Rule 6.3 holds trivially if rule 6.2 holds, and rule 6.4 holds straightforwardly by the description of $\mathcal{S}$.

Rule 7. This rule holds trivially by definition.

Rule 8. This rule holds as long as the differentiator cannot outputs $(SK \neq SK')$ such that $\mathbf{Gen}(SK) = \mathbf{Gen}(SK')$.

Rule 9. For this rule, it suffices to prove that for any V,

$$\mathbf{Verify}(PK, M, V) = 1 \Longleftrightarrow \mathcal{S}^{\Pi.\textsc{Verify}}(PK, M, V) = 1.$$

Sub-rule: $\mathbf{oVerify}(PK, M, V) = 1 \Longrightarrow \mathcal{S}^{\Pi.\textit{Verify}}(PK, M, V) = 1$. Given an V such that $\mathbf{Verify}(PK, M, V) = 1$, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{E}^{-1}}(V)$ as follows:

- Strategy 1. If $\exists(V, PK||M, \boldsymbol{v}) \in \mathbb{T}_{\mathcal{E}}$, then return $(\boldsymbol{v})$.
- Strategy 2. if $\exists(V, PK||M, \boldsymbol{v}, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $(\boldsymbol{v})$.
- Strategy 3. Otherwise, following the rest strategy in our simulator.

For strategy 3, trivial to note that, under the condition $\mathbf{Verify}(PK, M, V) = 1$, the response of $\mathcal{S}^{\mathcal{E}^{-1}}(V) = (\boldsymbol{v})$ is properly assigned such that $\mathcal{S}^{\Pi.\textsc{Verify}}(PK, M, V) = 1$ (same as the Rule 11 on page 21 in Sec 4). Moreover, we note that the table $\mathbb{T}_{\mathcal{E}^{-1}}$ only records the tuple such that $\mathbf{Verify}(PK, M, V) = 1$, thus the sub-rule also holds in strategy 2. And now, it's rest to prove the rule holds in strategy 1. In fact, there are three sub cases that the tuple $(V, PK||M, \boldsymbol{v}) \in \mathbb{T}_{\mathcal{E}}$,

1. $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ where $ctr_1 < n + 2$;
2. $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ where $ctr = n + 2$;
3. $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ where $\mathbf{Verify}(PK, M, V) = 0$.

In the first case, we have that $V \twoheadleftarrow \boldsymbol{\Sigma}$, which means that $\mathbf{Verify}(PK, M, V) = 0$ except for negligible probability ($\leq \frac{q|\mathcal{SK}|}{\boldsymbol{\Sigma}}$). For the second one, we have that $ctr = n + 2$ (case 2 in the yellow box or case 3 in the green box), referring to $\mathcal{S}^{\Pi.\textsc{Verify}}(PK, M, V) = 1$. For the last one, we have that $\mathbf{Verify}(PK, M, V) = 0$. Thus this sub-rule holds except for negligible probability.

Sub-rule: $\mathbf{Verify}(PK, M, V) = 1 \Longleftarrow \mathcal{S}^{\Pi.\textit{Verify}}(PK, M, V) = 1$. Given an V such that $\mathbf{Verify}(PK, M, V) = 1$, $\mathcal{S}$ responds to the query $\mathcal{S}^{\mathcal{E}^{-1}}(V)$ as follows:

- Strategy 1. If $\exists(V, PK||M, \boldsymbol{v}) \in \mathbb{T}_{\mathcal{E}}$, then return $(\boldsymbol{v})$.
- Strategy 2. if $\exists(V, PK||M, \boldsymbol{v}, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $(\boldsymbol{v})$.
- Strategy 3. Otherwise, following the rest strategy in our simulator.

Immediately observe that this sub-rule holds in strategy 2, as $\mathbb{T}_{\mathcal{E}^{-1}}$ only records tuples such that $\mathbf{Verify}(PK, M, V) = 1$. For strategy 3, we note that if $\mathbf{Verify}(PK, M, V) = 0$, then the response of $\mathcal{S}^{\mathcal{E}^{-1}}(V) = (\boldsymbol{v})$ is uniformly sampled, which means $\mathcal{S}^{\Pi.\textsc{Verify}}(PK, M, V) = 0$ except for negligible probability ($\leq \frac{1}{2^t}$).

Next we show that this sub rule also holds for strategy 1. Concretely, there are three sub cases that the tuple $(V, PK||M, \boldsymbol{v}) \in \mathbb{T}_{\mathcal{E}}$,

1. $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ in case 1 (red box);
2. $\mathcal{S}^{\mathcal{E}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ in case 2 (yellow box) or case 3 (green box);

3. $\mathcal{S}^{\mathcal{E}^{-1}}$ inserts $(V, PK||M, \boldsymbol{v})$ into $\mathbb{T}_{\mathcal{E}}$ where $\mathbf{Verify}(PK, M, V) = 0$.

For the second sub case, $V \leftarrow \mathbf{Sign}(SK, M, R)$ (either case 2 in the yellow box or case 3 in the green box), which means $\mathbf{Verify}(PK, M, V) = 1$. For the last one, as $(\boldsymbol{v})$ is uniformly sampled, which means $\mathcal{S}^{\Pi.\text{VERIFY}}(PK, M, V) = 0$ except for negligible probability. Thus this sub-rule holds except for negligible probability. Now it is rest to prove the first sub case, which includes several events:

1. $pad \neq 0 \cdots 0$;
2. $\exists SK \neq SK'$ s.t. $\mathbf{Gen}(SK) = \mathbf{Gen}(SK') = PK$;
3. $\nexists(SK, sk, PK) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}$;
4. $\nexists(PK, M, seed, b_1, \ldots, b_{n+1}) \in \mathbb{T}_{\mathcal{H}_{\text{path}}}$;
5. $\nexists(PK, M, seed, m) \in \mathbb{T}_{\mathcal{H}_{\text{msgLeaf}}}$
6. $0 < ctr_1 < n + 2$.

Trivial to note that, if $pad \neq 0 \cdots 0$ or $0 < ctr_1 < n + 2$, then $\mathcal{S}^{\Pi.\text{VERIFY}}(PK, M, V) = 0$. The second event is trivially bounded by $\frac{q^2}{|\mathcal{PK}|}$. Moreover, for the fourth and fifth event, the differentiator has no knowledge of either $b_1, \ldots, b_{n+1}$ or m, thus it's apparent that

$$\Pr[\mathcal{S}^{\Pi.\text{VERIFY}}(PK, M, V) = 1] \leq \frac{q}{2^n} + \frac{q}{|\mathbf{\Sigma}_{\mathsf{o}}|}.$$

For the third event, note that the differentiator could choose SK itself, but sk is independent of the differentiator's view. Hence, $\mathcal{D}$ can not output a valid v_0 via the adversarial interface directly; the only way it can extract valid v_0 is via honest interface. Specifically, $\mathcal{D}$ makes queries $V_i \leftarrow \mathbf{Sign}(SK, M_i, R_i)$ and $\boldsymbol{v}_i \leftarrow \mathcal{S}^{\mathcal{E}^{-1}}(V_i)$, and $\mathcal{D}$ can extract valid v_0^* from $\boldsymbol{v}_i$. However, for any M_i, we denote $b_1^i \cdots b_{n+1}^i \leftarrow \mathcal{S}^{\mathcal{H}_{\text{path}}}(PK, M_i, seed_i)$, and we note that except for negligible probability ($\leq \frac{q}{2^n}$), the most significant differ bit of $(b_1, \ldots, b_{n+1}, b_1^i \cdots b_{n+1}^i) < n$. As a result, the differentiator, although it can extract a valid v_0, it cannot outputs a sequence of valid signature $(v_0, \ldots, v_{n+1})$, which means that $\mathcal{S}^{\Pi.\text{VERIFY}}(PK, M, V) = 0$.

References

[BF18] Manuel Barbosa and Pooya Farshim. Indifferentiable authenticated encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220. Springer, Heidelberg, August 2018.

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *International Workshop on Fast Software Encryption*, pages 328–340. Springer, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

[CHK$^+$16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

[DKT16] Dana Dachman-Soled, Jonathan Katz, and Aishwarya Thiruvengadam. 10-round Feistel is indifferentiable from an ideal cipher. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 649–678. Springer, Heidelberg, May 2016.

[DP06] Yevgeniy Dodis and Prashant Puniya. On the relation between the ideal cipher and the random oracle models. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 184–206. Springer, Heidelberg, March 2006.

[DP07] Yevgeniy Dodis and Prashant Puniya. Feistel networks made public, and applications. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 534–554. Springer, Heidelberg, May 2007.

[DS16] Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round Feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Heidelberg, August 2016.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[Mau02] Ueli M. Maurer. Indistinguishability of random systems. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 110–132. Springer, Heidelberg, April / May 2002.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[ZZ20] Mark Zhandry and Cong Zhang. Indifferentiability for public key cryptosystems. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 63–93. Springer, Heidelberg, August 2020.

A Proof of Theorem 4.1

In this part, we complete the proof of Theorem 4.1, by illustrating a sequence of hybrid games, $\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}}, \ldots, \mathcal{G}_{\mathcal{S}_9}^{\mathcal{F}_9,\mathcal{D}}$, and proving that:

$$\Pr[\mathbf{Real}^{\Pi_\circ,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1],$$
$$\left|\Pr[\mathcal{G}_{\mathcal{S}_i}^{\mathcal{F}_i,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_{i+1}}^{\mathcal{F}_{i+1},\mathcal{D}} = 1]\right| \le \mathrm{negl}(\lambda),$$
$$\Pr[\mathcal{G}_{\mathcal{S}_9}^{\mathcal{F}_9,\mathcal{D}} = 1] = \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{OSIG},\mathcal{D}}]$$

where $\mathcal{F}_i$ and $\mathcal{S}_i$ are the corresponding ideal functionality and simulator in each hybrid game, respectively.

The functionalities $\mathcal{F}_i$ for $i = 0, 1 \ldots 8$, are all the same, a "dummy functionality" which forward the queries from the differentiator $\mathcal{D}$ to the simulator $\mathcal{S}_i$, and vice versa. The functionality $\mathcal{F}_9$ is the same as the ideal signature **OSIG**. In the remaining, we carefully illustrate the description of the simulators $\mathcal{S}_i$, for each hybrid game, and then prove that the statistical distance of each two adjacent games are close.

Simulator $\mathcal{S}_0$

The simulator $\mathcal{S}_0$ will provide internal copies of the random oracles, $\mathcal{H}_{\mathrm{sk}}$, $\mathcal{H}_{\mathrm{OneWay}}$, $\mathcal{H}_{\mathrm{position}}$, and random permutation ($\mathcal{P}$, $\mathcal{P}^{-1}$), and ideal cipher ($\mathcal{E}$, $\mathcal{E}^{-1}$), and internal copy of $\Pi_\circ$ =$\Pi_\circ$.{oGen, oSign, oVerify }; the simulator $\mathcal{S}_0$ has the external oracle access to $\mathcal{F}_0$; the simulator $\mathcal{S}_0$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\underline{\mathcal{S}_0^{\mathcal{H}_{\mathrm{sk}}}(SK_\circ)}$:

$\boldsymbol{sk} \leftarrow \mathcal{H}_{\mathrm{sk}}(SK_\circ)$, return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_0^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk})}$:

$\widehat{pk} \leftarrow \mathcal{H}_{\mathrm{OneWay}}(\widehat{sk})$, return $\widehat{pk}$.

$\underline{\mathcal{S}_0^{\mathcal{H}_{\mathrm{position}}}(PK_\circ, M_\circ)}$:

$b_1 \cdots b_n \leftarrow \mathcal{H}_{\mathrm{position}}(PK_\circ, M_\circ)$, return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_0^{\mathcal{P}}(\boldsymbol{pk})}$:

$PK_\circ \leftarrow \mathcal{P}(\boldsymbol{pk})$, return $PK_\circ$.

$\underline{\mathcal{S}_0^{\mathcal{P}^{-1}}(PK_\circ)}$:

$\boldsymbol{pk} \leftarrow \mathcal{P}^{-1}(PK_\circ)$, return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_0^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:

$V_\circ \leftarrow \mathcal{E}(PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$, return $V_\circ$.

$\underline{\mathcal{S}_0^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)}$:

$(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad}) \leftarrow \mathcal{E}^{-1}(PK_\circ || M_\circ, V_\circ)$, return $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})$.

It is straightforward that the view of $\mathcal{D}$ in either **Real** or $\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}}$ are identical, thus

$$\Pr[\mathbf{Real}^{\Pi_\circ,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1].$$

Next, we will present the description of the other hybrids.

Simulator $\mathcal{S}_1$

The simulator $\mathcal{S}_1$ will provide internal copies of the random oracles, $\mathcal{H}_{\mathrm{sk}}$, $\mathcal{H}_{\mathrm{OneWay}}$, $\mathcal{H}_{\mathrm{position}}$, and random permutation ($\mathcal{P}$, $\mathcal{P}^{-1}$), and ideal cipher ($\mathcal{E}$, $\mathcal{E}^{-1}$), and internal copy of $\Pi_\circ$ =$\Pi_\circ$.{oGen, oSign, oVerify }; the simulator $\mathcal{S}_1$ has the external oracle access to $\mathcal{F}_1$; the simulator $\mathcal{S}_1$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\underline{\mathcal{S}_1^{\mathcal{H}_{\mathrm{sk}}}(SK_\circ)}$:

```
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGen}(SK_\circ)$
    then $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$,
    return $\boldsymbol{sk}$;
$\boldsymbol{sk} \leftarrow \mathcal{H}_{\mathrm{sk}}(SK_\circ)$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_1^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_1^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_1^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk})}$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\mathrm{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\underline{\mathcal{S}_1^{\mathcal{H}_{\mathrm{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\mathrm{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_1^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $PK_\circ$;
$PK_\circ \leftarrow \mathcal{P}(\boldsymbol{pk})$, $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(\diamond, \diamond, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_1^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{pk} \leftarrow \mathcal{P}^{-1}(PK_\circ)$; $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(\diamond, \diamond, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_1^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
```

```
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK}_\circ \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1: $V_\circ \leftarrow \mathcal{E}(PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2: $V_\circ \leftarrow \Pi_\circ.\text{oSIGN}(\widehat{SK}_\circ, M_\circ)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.

$\mathcal{S}_1^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\text{oVERIFY}(PK_\circ || M_\circ, V_\circ) = 0$,
    then $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad}) \leftarrow \mathcal{E}^{-1}(PK_\circ, M_\circ, V_\circ)$,
    $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})\}$,
    return $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad}) \leftarrow \mathcal{E}^{-1}(PK_\circ, M_\circ, V_\circ)$,
    $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})\}$,
    return $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad}) \leftarrow \mathcal{E}^{-1}(PK_\circ, M_\circ, V_\circ)$,
    $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})\}$,
    return $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})$.
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Comparing to $\mathcal{S}_0$, the simulator $\mathcal{S}_1$ keeps several tables and has additional oracles ($\Pi_\circ$.oGEN, $\Pi_\circ$.oSIGN, $\Pi_\circ$.oVERIFY). When $\mathcal{S}_1$ responds to a query, it first checks the tables, if the proper response is recorded in one of the tables, then

$\mathcal{S}_1$ responds to the query using the table, else $\mathcal{S}_1$ responds to the query by calling oracles. In the following, we prove that, for any query, the response of $\mathcal{S}_0$ and $\mathcal{S}_1$ is identical with high probability, which straightforwardly refers to that

$$\big|\Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_1}^{\mathcal{F}_1,\mathcal{D}} = 1]\big| \le \text{negl}(\lambda).$$

In Game 0, $\mathcal{S}_0^{\mathcal{H}_{\text{sk}}}(SK_\circ) = \mathcal{H}_{\text{sk}}(SK_\circ)$, and in Game 1, $\mathcal{S}_1$ responds to this query either using its table $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$ or calling $\mathcal{H}_{\text{sk}}(\cdot)$. It's trivial that $\mathcal{S}_0^{\mathcal{H}_{\text{sk}}}(SK_\circ) = \mathcal{S}_1^{\mathcal{H}_{\text{sk}}}(SK_\circ)$ if $\mathcal{S}_1$ responds to the query by calling $\mathcal{H}_{\text{sk}}(SK_\circ)$, and next we analyze the case where $\mathcal{S}_1$ responds to query by using table $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$. Note that in Game 1, none of the sub-simulator will insert tuple with form of $(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)$ into $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$, and only $\mathcal{S}_1^{\mathcal{H}_{\text{sk}}}$ might insert tuple $(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)$ into $\mathbb{T}_{\mathcal{H}_{\text{sk}}}$. Moreover, the tuple inserted by $\mathcal{S}_1^{\mathcal{H}_{\text{sk}}}$ is consistent with the responses of oracles. More concretely, if $(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)$ is inserted, then it's apparent that

$$\boldsymbol{sk} = \mathcal{H}_{\text{sk}}(SK_\circ), \boldsymbol{pk} = \mathcal{H}_{\text{OneWay}}(\boldsymbol{sk}), PK_\circ = \mathcal{P}(\boldsymbol{pk}).$$

Thus, we have that $\mathcal{S}_0^{\mathcal{H}_{\text{sk}}}(SK_\circ) = \mathcal{S}_1^{\mathcal{H}_{\text{sk}}}(SK_\circ)$. Applying the same analysis, we can straightforwardly argue that $\mathcal{S}_0^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}) = \mathcal{S}_1^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}), \mathcal{S}_0^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ) = \mathcal{S}_1^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ), \mathcal{S}_0^{\mathcal{P}}(\boldsymbol{pk}) = \mathcal{S}_1^{\mathcal{P}}(\boldsymbol{pk})$, and $\mathcal{S}_0^{\mathcal{P}^{-1}}(PK_\circ) = \mathcal{S}_1^{\mathcal{P}^{-1}}(PK_\circ)$. Next, we prove that this equivalence also holds in $\mathcal{E}$ and $\mathcal{E}^{-1}$. In Game 0, we have that

$$\mathcal{S}_0^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad) = \mathcal{E}(PK_\circ||M_\circ, sk_{1,b_1}||\cdots||sk_{n,b_n}||pad),$$

and in Game 1, $\mathcal{S}_1$ responds to this query in the following cases: 1) using its table $\mathbb{T}_{\mathcal{E}}$; 2) using its table $\mathbb{T}_{\mathcal{E}^{-1}}$; 3) calling $\mathcal{E}(\cdot,\cdot)$; 4) returning $\Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ)$. Trivial to note that, the tuples recorded in tables are consistent with the oracles, and thus it suffices to analyze the last case. By definition, we have that $\mathcal{S}_1^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad) = \Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ)$ if and only if the following conditions hold:

1. $pad = 0\cdots 0$;
2. there is a tuple $(\widehat{SK_\circ}, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$;
3. there is a tuple $(PK_\circ, M_\circ, \widehat{b}_1, \dots, \widehat{b}_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$;
4. $sk_{i,b_i} = \widehat{sk}_{i,\widehat{b}_i}$ for $i \in [1, n]$.

Trivial to note that, if those four conditions hold, then we have that

$$\begin{aligned}\Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ) &= \mathcal{E}(PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}||\cdots||\widehat{sk}_{n,\widehat{b}_n}||0\cdots 0)\\ &= \mathcal{E}(PK_\circ||M_\circ, sk_{1,b_1}||\cdots||sk_{n,b_n}||pad).\end{aligned}$$

Thus we have that

$$\mathcal{S}_0^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad) = \mathcal{S}_1^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad).$$

And for $\mathcal{E}^{-1}$, we have that, in Game 0, $\mathcal{S}_0^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ) = \mathcal{E}^{-1}(PK_\circ||M_\circ, V_\circ)$. While, in Game 1, $\mathcal{S}_1$ responds to this query in the following cases: 1) using its table $\mathbb{T}_{\mathcal{E}}$; using its table $\mathbb{T}_{\mathcal{E}^{-1}}$; 3) calling $\mathcal{E}^{-1}(\cdot,\cdot)$; 4) returning $(\widehat{sk}_{1,\widehat{b}_1}, \dots, \widehat{sk}_{n,\widehat{b}_n}, 0\cdots 0)$. Trivial to note that $\mathcal{S}_0^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ) = \mathcal{S}_1^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)$ if $\mathcal{S}_1$ responds to the query within the first three cases, and for the last one, we have that $\mathcal{S}_1^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ) = (\widehat{sk}_{1,\widehat{b}_1}, \dots, \widehat{sk}_{n,\widehat{b}_n}, 0\cdots 0)$ if and only if the following conditions hold:

1. $\Pi_\circ.\text{oVerify}(PK_\circ||M_\circ, V_\circ) = 1$;
2. there exist no $(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$;
3. there is a tuple $(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$;
4. there is a tuple $(PK_\circ, M_\circ, \widehat{b}_1, \dots, \widehat{b}_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$;

5. $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,\widehat{b}_i}$ for $i \in [1, n]$.

Note that if $V_\circ = \Pi_\circ.\text{oSIGN}(SK_\circ, M_\circ)$, then it's straightforward that $\mathcal{S}_1^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ) = \mathcal{E}^{-1}(PK_\circ||M_\circ, V_\circ)$. And thus the only bad event is: $\mathcal{D}$ outputs $V_\circ \neq \text{oSIGN}(SK_\circ, M_\circ)$ such that $\Pi_\circ.\text{oVERIFY}(PK_\circ||M_\circ, V_\circ) = 1$, and we then prove that this bad event would never occur except for negligible probability.

With loss of generality, we assume that the adversary makes q_1 queries via the honest interfaces and q_2 queries via the adversarial interfaces, where $q_1 + q_2 \leq q$. Note that, $V_\circ \notin \mathbb{T}_\mathcal{E} \cup \mathbb{T}_{\mathcal{E}^{-1}}$, thus, we know that $V_\circ$ never appears in the queries via the adversarial interfaces. Therefore, $V_\circ$ either is randomly guessed or appears in the queries via the honest interfaces. For the former case, trivial to note that the probability that $\Pi_\circ.\text{oVERIFY}(PK_\circ||M_\circ, V_\circ) = 1$ is bounded by $\frac{1}{2^{n\lambda}}$. For the latter case, we assume that $V_\circ = \text{oSIGN}(\widehat{SK}_\circ, M_\circ)$. If $\Pi_\circ.\text{oGEN}(\widehat{SK}_\circ) \neq PK_\circ$, then the probability that $\Pi_\circ.\text{oVERIFY}(PK_\circ||M_\circ, V_\circ) = 1$ is bounded by $\frac{1}{2^{n\lambda}}$. If $\Pi_\circ.\text{oGEN}(\widehat{SK}_\circ) = PK_\circ$, then the probability that $\Pi_\circ.\text{oVERIFY}(PK_\circ||M_\circ, V_\circ) = 1$ is bounded by the event that outputs $SK_\circ \neq SK'_\circ$ such that $\Pi_\circ.\text{oGEN}(SK_\circ) = \Pi_\circ.\text{oGEN}(SK'_\circ)$, which is $\frac{q_1^2}{2^{2n\lambda}}$. Combine together, we have that:

$$\left|\Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_1}^{\mathcal{F}_1,\mathcal{D}} = 1]\right| \leq \frac{2q}{2^{n\lambda}} + \frac{2q^3}{2^{2n\lambda}}.$$

Next, we give the description of $\mathcal{S}_2$.

Simulator $\mathcal{S}_2$

The simulator $\mathcal{S}_2$ will provide internal copies of the random oracles, $\mathcal{H}_{\text{sk}}$, $\mathcal{H}_{\text{OneWay}}$, $\mathcal{H}_{\text{position}}$, and random permutation ($\mathcal{P}$, $\mathcal{P}^{-1}$), and ideal cipher $\mathcal{E}$, and internal copy of $\Pi_\circ = \Pi_\circ.\{\text{oGEN}, \text{oSIGN}, \text{oVERIFY}\}$; the simulator $\mathcal{S}_2$ has the external oracle access to $\mathcal{F}_2$; the simulator $\mathcal{S}_2$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_2^{\mathcal{H}_{\text{sk}}}(SK_\circ)$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGEN}(SK_\circ)$
    then $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$,
    return $\boldsymbol{sk}$;
$\boldsymbol{sk} \leftarrow \mathcal{H}_{\text{sk}}(SK_\circ)$,$PK_\circ \leftarrow \Pi_\circ.\text{oGEN}(SK_\circ)$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_2^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_2^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\mathcal{S}_2^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\text{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\mathcal{S}_2^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.
```

```
$\underline{\mathcal{S}_2^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
$PK_\circ \leftarrow \mathcal{P}(\boldsymbol{pk}), \mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \diamond, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_2^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{pk} \leftarrow \mathcal{P}^{-1}(PK_\circ); \mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \diamond, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_2^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1} \rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1} \rangle\Big)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \leftarrow \mathcal{E}(PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.

$\underline{\mathcal{S}_2^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$, then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\text{oVerify}(PK_\circ || M_\circ, V_\circ) = 0$,  //for invalid signatures, $\mathcal{S}_2$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
    $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ || M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
    return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$, //for this bad event, $\mathcal{S}_2$ responds with random strings
  then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
    for $i \in [1, n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
    $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ || M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
    return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists (PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
  then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
  else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
  then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists (\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
  then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
  then $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad}) \leftarrow \mathcal{E}^{-1}(PK_\circ, M_\circ, V_\circ)$,
  $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})\}$,
  return $(\widehat{sk}_{1,b_1}, \ldots, \widehat{sk}_{n,b_n}, \widehat{pad})$.
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next, we give the description of $\mathcal{S}_3$.

Simulator $\mathcal{S}_3$

The simulator $\mathcal{S}_3$ will provide internal copies of the random oracles, $\mathcal{H}_{\text{sk}}$, $\mathcal{H}_{\text{OneWay}}$, $\mathcal{H}_{\text{position}}$, and random permutation $\mathcal{P}$, and ideal cipher $\mathcal{E}$, and internal copy of $\Pi_\circ = \Pi_\circ.\{\text{oGen}, \text{oSign}, \text{oVerify}\}$; the simulator $\mathcal{S}_3$ has the external oracle access to $\mathcal{F}_3$; the simulator $\mathcal{S}_3$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_3^{\mathcal{H}_{\text{sk}}}(SK_\circ)}$:
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
  then return $\boldsymbol{sk}$;
if $\exists (\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGen}(SK_\circ)$
  then $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$,
  return $\boldsymbol{sk}$;
$\boldsymbol{sk} \leftarrow \mathcal{H}_{\text{sk}}(SK_\circ)$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$,
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
  $pk_{i,0} = \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
  $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})}$:
```

```
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\text{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\underline{\mathcal{S}_3^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_3^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
$PK_\circ \leftarrow \mathcal{P}(\boldsymbol{pk})$, $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \diamond, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_3^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    $pk_{i,0} \leftarrow \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_3^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
```

```
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \dots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,   //check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \leftarrow \mathcal{E}(PK_\circ, M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\text{oSign}(\widehat{SK}_\circ, M_\circ)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad)\}$; return $V_\circ$.

$\mathcal{S}_3^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)$:
if $\exists (V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(sk_{1,b_1}, \dots, sk_{n,b_n}, pad)$.
if $\exists (V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \dots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \dots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\text{oVerify}(PK_\circ || M_\circ, V_\circ) = 0$,   //for invalid signatures, $\mathcal{S}_3$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ || M_\circ, \widehat{sk}_1, \dots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \dots, \widehat{sk}_n, \widehat{pad})$.
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,   //for this bad event, $\mathcal{S}_3$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ || M_\circ, \widehat{sk}_1, \dots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \dots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists (PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists (\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
    parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \dots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
    for $i \in [1, n]$
        $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,0})$, $\widehat{pk}_{i,1} \leftarrow \mathcal{S}_3^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,1})$,
        $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \dots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
    $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \dots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \dots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$; $\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ || M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \dots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
return $(\widehat{sk}_{1,\widehat{b}_1}, \dots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next, we give the description of $\mathcal{S}_4$.

Simulator $\mathcal{S}_4$

The simulator $\mathcal{S}_4$ will provide internal copies of the random oracles, $\mathcal{H}_{\text{sk}}$, $\mathcal{H}_{\text{OneWay}}$, $\mathcal{H}_{\text{position}}$, and ideal cipher $\mathcal{E}$, and internal copy of $\Pi_\circ$ =$\Pi_\circ$.{oGen, oSign, oVerify }; the simulator $\mathcal{S}_4$ has the external oracle access to $\mathcal{F}_4$; the simulator $\mathcal{S}_4$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_4^{\mathcal{H}_{\text{sk}}}(SK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGen}(SK_\circ)$
    then $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$,
    return $\boldsymbol{sk}$;
$\boldsymbol{sk} \leftarrow \mathcal{H}_{\text{sk}}(SK_\circ)$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$,
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_4^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_4^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
    return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_4^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})}$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\text{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.
$\underline{\mathcal{S}_4^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_4^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1, n]$ ,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,   //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,   //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}, \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\mathrm{OGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_4^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \diamond, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $pk_{i,0} \leftarrow \mathcal{S}_4^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_4^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_4^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then goto Case 1;
    else $\widehat{SK}_\circ \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1: $V_\circ \leftarrow \mathcal{E}(PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2: $V_\circ \leftarrow \Pi_\circ.\mathrm{oSign}(\widehat{SK}_\circ, M_\circ)$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
```

```
$\underline{\mathcal{S}_4^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)}$:
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ$.oVerify$(PK_\circ||M_\circ, V_\circ) = 0$,  //for invalid signatures, $\mathcal{S}_4$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$, //for this bad event, $\mathcal{S}_4$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$;
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\mathrm{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
    parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
    for $i \in [1, n]$
        $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_4^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk}_{i,0}), \widehat{pk}_{i,1} \leftarrow \mathcal{S}_4^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk}_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
    $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$; parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$; $\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$; return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next we give the description of $\mathcal{S}_5$.

Simulator $\mathcal{S}_5$

The simulator $\mathcal{S}_5$ will provide internal copies of the random oracles, $\mathcal{H}_{\mathrm{sk}}$, $\mathcal{H}_{\mathrm{OneWay}}$, $\mathcal{H}_{\mathrm{position}}$, and internal copy of $\Pi_\circ$ =$\Pi_\circ$.{oGen, oSign, oVerify }; the simulator $\mathcal{S}_5$ has the external oracle access to $\mathcal{F}_5$; the simulator $\mathcal{S}_5$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_5^{\mathcal{H}_{\mathrm{sk}}}(SK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$, then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$ s.t. $PK_\circ = \Pi_\circ$.oGen$(SK_\circ)$ then $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$; return $\boldsymbol{sk}$;
$\boldsymbol{sk} \leftarrow \mathcal{H}_{\mathrm{sk}}(SK_\circ)$, $PK_\circ \leftarrow \Pi_\circ$.oGen$(SK_\circ)$; parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_5^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,0}), pk_{i,1} = \mathcal{S}_5^{\mathcal{H}_{\mathrm{OneWay}}}(sk_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.
```

```
$\underline{\mathcal{S}_5^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})}$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\text{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\underline{\mathcal{S}_5^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_5^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1, n]$,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\textsc{oGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_5^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    $pk_{i,0} \leftarrow \mathcal{S}_5^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_5^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_5^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
```

```
if $pad \neq 0\cdots0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,  //check the validity of each $sk_{i,b_i}$
      then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \twoheadleftarrow \boldsymbol{\Sigma}_\circ$; $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ)$;
         $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.

$\underline{\mathcal{S}_5^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)}$:
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_\mathcal{E}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\text{oVerify}(PK_\circ||M_\circ, V_\circ) = 0$,  //for invalid signatures, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
      for $i \in [1, n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
      $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
      return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,  //for this bad event, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
      for $i \in [1, n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
      $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
      return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$;
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$; parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
    for $i \in [1, n]$
      $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_5^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,0})$, $\widehat{pk}_{i,1} \leftarrow \mathcal{S}_5^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
    $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$; parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0\cdots0$; $\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$; return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next, we give the description of $\mathcal{S}_6$.

Simulator $\mathcal{S}_6$

The simulator $\mathcal{S}_6$ will provide internal copies of the random oracles, $\mathcal{H}_{\text{OneWay}}$, $\mathcal{H}_{\text{position}}$, and internal copy of $\Pi_\circ = \Pi_\circ.\{$oGen, oSign, oVerify $\}$; the simulator $\mathcal{S}_6$ has the external oracle access to $\mathcal{F}_6$; the simulator $\mathcal{S}_6$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_6^{\mathcal{H}_{\text{sk}}}(SK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGen}(SK_\circ)$
    then $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$,
    return $\boldsymbol{sk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$,
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_6^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_6^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_6^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})}$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \mathcal{H}_{\text{OneWay}}(\widehat{sk})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\underline{\mathcal{S}_6^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_6^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1, n]$,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\textsc{oGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.
```

$\underline{\mathcal{S}_6^{\mathcal{P}^{-1}}(PK_\circ)}$:

```
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $pk_{i,0} \leftarrow \mathcal{S}_6^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_6^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.
```

$\underline{\mathcal{S}_6^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:

```
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0\cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK}_\circ \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^*\cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1\cdots\widehat{b}_n \leftarrow b_1^*\cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \twoheadleftarrow \boldsymbol{\Sigma}_\circ$; $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\textsc{oSign}(\widehat{SK}_\circ, M_\circ)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
```

$\underline{\mathcal{S}_6^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)}$:

```
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\textsc{oVerify}(PK_\circ||M_\circ, V_\circ) = 0$,  //for invalid signatures, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1,n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
      $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
      return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk'}, \boldsymbol{pk'}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$, //for this bad event, $\mathcal{S}_5$ responds with random strings
   then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
      for $i \in [1, n]$
        $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
      $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
      return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists (PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\rm position}}$,
   then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
   else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\rm position}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\rm position}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm position}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$,
   then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists (\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$,
   then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
   then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
   parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
   for $i \in [1, n]$
      $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_6^{\mathcal{H}_{\rm OneWay}}(\widehat{sk}_{i,0})$, $\widehat{pk}_{i,1} \leftarrow \mathcal{S}_6^{\mathcal{H}_{\rm OneWay}}(\widehat{sk}_{i,1})$,
      $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
   $\widehat{\boldsymbol{pk}} \leftarrow \Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
   $\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next we give the description of $\mathcal{S}_7$.

Simulator $\mathcal{S}_7$

The simulator $\mathcal{S}_7$ will provide internal copies of the random oracle $\mathcal{H}_{\rm position}$, and internal copy of $\Pi_\circ = \Pi_\circ.\{$oGen, oSign, oVerify$\}$; the simulator $\mathcal{S}_7$ has the external oracle access to $\mathcal{F}_7$; the simulator $\mathcal{S}_7$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_7^{\mathcal{H}_{\rm sk}}(SK_\circ)$:
if $\exists (SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$, then return $\boldsymbol{sk}$;
if $\exists (\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$ s.t. $PK_\circ = \Pi_\circ.$oGen$(SK_\circ)$ then $\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$; return $\boldsymbol{sk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.$oGen$(SK_\circ)$; parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
   $pk_{i,0} = \mathcal{S}_7^{\mathcal{H}_{\rm OneWay}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_7^{\mathcal{H}_{\rm OneWay}}(sk_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$; $\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\mathcal{S}_7^{\mathcal{H}_{\rm OneWay}}(\widehat{sk})$:
if $\exists (\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$,
   then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \{0,1\}^\lambda$; $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
```

```
return $\widehat{pk}$.

$\underline{\mathcal{S}_7^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1\cdots b_n$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1},\ldots,sk_{n,b_n}, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1\cdots b_n$;
$b_1\cdots b_n \leftarrow \mathcal{H}_{\text{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1\cdots b_n)\}$;
return $b_1\cdots b_n$.

$\underline{\mathcal{S}_7^{\mathcal{P}}(\boldsymbol{pk})}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle,\ldots,\langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1,n]$,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,   //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,   //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle,\ldots,\langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\underline{\mathcal{S}_7^{\mathcal{P}^{-1}}(PK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle,\ldots,\langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $pk_{i,0} \leftarrow \mathcal{S}_7^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_7^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle,\ldots,\langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_7^{\mathcal{E}}(PK_\circ||M_\circ, sk_{1,b_1},\ldots,sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1},\ldots,sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1},\ldots,sk_{1,b_n}, pad, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0\cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
```

```
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\left(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\right)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \twoheadleftarrow \boldsymbol{\Sigma}_\circ$; $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\mathrm{oSign}(\widehat{SK_\circ}, M_\circ)$;
         $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
```

```
$\underline{\mathcal{S}_7^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)}$:
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_\mathcal{E}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\mathrm{oVerify}(PK_\circ||M_\circ, V_\circ) = 0$,  //for invalid signatures, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$,  //for this bad event, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\mathrm{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow \mathcal{H}_{\mathrm{position}}(PK_\circ, M_\circ)$; $\mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$, then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}}$, then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$; parse $\widehat{\boldsymbol{sk}}$ into $\left(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\right)$;
    for $i \in [1, n]$
        $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_7^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk}_{i,0})$, $\widehat{pk}_{i,1} \leftarrow \mathcal{S}_7^{\mathcal{H}_{\mathrm{OneWay}}}(\widehat{sk}_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \left(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\right)$;
    $\mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\mathrm{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\left(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\right)$;
parse $\widehat{\boldsymbol{pk}}$ into $\left(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\right)$;
$\widehat{pad} \leftarrow 0 \cdots 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
```

```
return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next, we give the description of $\mathcal{S}_8$.

Simulator $\mathcal{S}_8$

The simulator $\mathcal{S}_8$ will provide internal copies of the internal copy of $\Pi_\circ = \Pi_\circ.\{\text{oGen}, \text{oSign}, \text{oVerify}\}$; the simulator $\mathcal{S}_8$ has the external oracle access to $\mathcal{F}_8$; the simulator $\mathcal{S}_8$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_8^{\mathcal{H}_{\rm sk}}(SK_\circ)$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$, then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$ s.t. $PK_\circ = \Pi_\circ.\text{oGen}(SK_\circ)$, then $\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$; return $\boldsymbol{sk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$; parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1,n]$
    $pk_{i,0} = \mathcal{S}_8^{\mathcal{H}_{\rm OneWay}}(sk_{i,0})$, $pk_{i,1} = \mathcal{S}_8^{\mathcal{H}_{\rm OneWay}}(sk_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$; $\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$; return $\boldsymbol{sk}$.

$\mathcal{S}_8^{\mathcal{H}_{\rm OneWay}}(\widehat{sk})$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$, then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \{0,1\}^{\lambda}$; $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\mathcal{S}_8^{\mathcal{H}_{\rm position}}(PK_\circ, M_\circ)$:
if $\exists(PK_\circ, M_\circ, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\rm position}}$,
    then return $b_1\cdots b_n$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1\cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1\cdots b_n$;
$b_1\cdots b_n \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\rm position}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm position}} \cup \{(PK_\circ, M_\circ, b_1\cdots b_n)\}$;
return $b_1\cdots b_n$.

$\mathcal{S}_8^{\mathcal{P}}(\boldsymbol{pk})$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\rm sk}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1,n]$,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$,   //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$,   //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\rm OneWay}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\rm OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm OneWay}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \Pi_\circ.\text{oGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\rm sk}} \leftarrow \mathbb{T}_{\mathcal{H}_{\rm sk}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.

$\mathcal{S}_8^{\mathcal{P}^{-1}}(PK_\circ)$:
```

```
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    $pk_{i,0} \leftarrow \mathcal{S}_8^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0}), pk_{i,1} \leftarrow \mathcal{S}_8^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\underline{\mathcal{S}_8^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1,n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,   //check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \twoheadleftarrow \boldsymbol{\Sigma}_\circ$; $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \Pi_\circ.\text{oSign}(\widehat{SK_\circ}, M_\circ)$;
         $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.

$\underline{\mathcal{S}_8^{\mathcal{E}^{-1}}(PK_\circ || M_\circ, V_\circ)}$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\Pi_\circ.\text{oVerify}(PK_\circ || M_\circ, V_\circ) = 0$,   //for invalid signatures, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1,n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ || M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,   //for this bad event, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1,n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_{\mathcal{E}} \leftarrow \mathbb{T}_{\mathcal{E}} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
    parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
    for $i \in [1, n]$
        $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_8^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,0}), \widehat{pk}_{i,1} \leftarrow \mathcal{S}_8^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk}_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
    $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$; parse $\widehat{\boldsymbol{pk}}$ into $\Big(\langle \widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle \widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\Big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$; $\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$; return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

Next, we give the description of $\mathcal{S}_9$.

$\mathcal{S}_9$ with oracle access to (oGen, oSign, oVerify)

The simulator $\mathcal{S}_8$ has the oracle access to (**oGen**, **oSign**, **oVerify**); $\mathcal{S}_8$ will provide the following adversarial interfaces to the differentiator:

```
$\underline{\mathcal{S}_9^{\mathcal{H}_{\text{sk}}}(SK_\circ)}$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$, then return $\boldsymbol{sk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$ s.t. $PK_\circ = \textbf{oGen}(SK_\circ)$, then $\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$; return $\boldsymbol{sk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$, $PK_\circ \leftarrow \textbf{oGen}(SK_\circ)$; parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \ldots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$,
for $i \in [1, n]$
    $pk_{i,0} = \mathcal{S}_9^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$; $pk_{i,1} = \mathcal{S}_9^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \ldots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{sk}$.

$\underline{\mathcal{S}_9^{\mathcal{H}_{\text{OneWay}}}(\widehat{sk})}$:
if $\exists(\widehat{sk}, \widehat{pk}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
    then return $\widehat{pk}$;
$\widehat{pk} \leftarrow \{0,1\}^\lambda$; $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}, \widehat{pk})\}$;
return $\widehat{pk}$.

$\underline{\mathcal{S}_9^{\mathcal{H}_{\text{position}}}(PK_\circ, M_\circ)}$:
if $\exists(PK_\circ, M_\circ, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$,
    then return $b_1 \cdots b_n$;
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $b_1 \cdots b_n$;
$b_1 \cdots b_n \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_{\text{position}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{position}}} \cup \{(PK_\circ, M_\circ, b_1 \cdots b_n)\}$;
return $b_1 \cdots b_n$.

$\underline{\mathcal{S}_9^{\mathcal{P}}(\boldsymbol{pk})}$:
```

```
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $PK_\circ$;
parse $\boldsymbol{pk}$ into $\Big(\langle pk_{1,0}, pk_{1,1}\rangle, \dots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$,
for $i \in [1, n]$,
    if $\exists(sk_{i,0}, pk_{i,0}) \neq (sk'_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,0}$
        then return $\perp$;
    if $\exists(sk_{i,0}, pk_{i,0}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,0} \leftarrow sk_{i,0}$,
        else $\widehat{sk}_{i,0} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,0}, pk_{i,0})\}$
    if $\exists(sk_{i,1}, pk_{i,1}) \neq (sk'_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,  //collision on $pk_{i,1}$
        then return $\perp$;
    if $\exists(sk_{i,1}, pk_{i,1}) \in \mathbb{T}_{\mathcal{H}_{\text{OneWay}}}$,
        then $\widehat{sk}_{i,1} \leftarrow sk_{i,1}$,
        else $\widehat{sk}_{i,1} \twoheadleftarrow \{0,1\}^{2\lambda}$, $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(\widehat{sk}_{i,1}, pk_{i,1})\}$,
$\widehat{\boldsymbol{sk}} \leftarrow \Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \dots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$, $SK_\circ \twoheadleftarrow \{0,1\}^{8n\lambda}$, $PK_\circ \leftarrow \mathbf{oGen}(SK_\circ)$,
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(SK_\circ, \widehat{\boldsymbol{sk}}, \boldsymbol{pk}, PK_\circ)\}$,
return $PK_\circ$.


$\mathcal{S}_9^{\mathcal{P}^{-1}}(PK_\circ)$:
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then return $\boldsymbol{pk}$;
$\boldsymbol{sk} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
parse $\boldsymbol{sk}$ into $\Big(\langle sk_{1,0}, sk_{1,1}\rangle, \dots, \langle sk_{n,0}, sk_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    $pk_{i,0} \leftarrow \mathcal{S}_9^{\mathcal{H}_{\text{OneWay}}}(sk_{i,0})$, $pk_{i,1} \leftarrow \mathcal{S}_9^{\mathcal{H}_{\text{OneWay}}}(sk_{i,1})$,
    $\mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{OneWay}}} \cup \{(sk_{i,0}, pk_{i,0})\} \cup \{(sk_{i,1}, pk_{i,1})\}$;
$\boldsymbol{pk} \leftarrow \Big(\langle pk_{1,0}, pk_{1,1}\rangle, \dots, \langle pk_{n,0}, pk_{n,1}\rangle\Big)$;
$\mathbb{T}_{\mathcal{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathcal{H}_{\text{sk}}} \cup \{(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ)\}$;
return $\boldsymbol{pk}$.

$\mathcal{S}_9^{\mathcal{E}}(PK_\circ || M_\circ, sk_{1,b_1}, \dots, sk_{1,b_n}, pad)$:
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \dots, sk_{1,b_n}, pad) \in \mathbb{T}_{\mathcal{E}}$,
    then return $V_\circ$;
if $\exists(V_\circ, PK_\circ || M_\circ, sk_{1,b_1}, \dots, sk_{1,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $V_\circ$;
$ctr \leftarrow 0$;
if $pad \neq 0 \cdots 0$,
    then goto Case 1;
if $\exists(SK_\circ \neq SK'_\circ)$ s.t. $\Big((SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big) \wedge \Big((SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{skRoot}}}\Big)$
    then goto Case 1;
if $\nexists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_{\text{sk}}}$,
    then goto Case 1;
    else $\widehat{SK_\circ} \leftarrow SK_\circ$; $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$;
if $\nexists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_{\text{position}}}$
    then goto Case 1;
    else $\widehat{b}_1 \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
parse $\widehat{\boldsymbol{sk}}$ into $\Big(\langle \widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \dots, \langle \widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\Big)$;
for $i \in [1, n]$
    if $\widehat{sk}_{i,\widehat{b}_i} = sk_{i,b_i}$,
//check the validity of each $sk_{i,b_i}$
        then $ctr \leftarrow ctr + 1$;
if $ctr < n$
    then goto Case 1;
    else goto Case 2;
Case 1:  $V_\circ \twoheadleftarrow \mathbf{\Sigma}_\circ$; $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.
Case 2:  $V_\circ \leftarrow \mathbf{oSign}(\widehat{SK_\circ}, M_\circ)$ ;
         $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{(V_\circ, PK_\circ, M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)\}$; return $V_\circ$.


$\mathcal{S}_9^{\mathcal{E}^{-1}}(PK_\circ||M_\circ, V_\circ)$:
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad) \in \mathbb{T}_\mathcal{E}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\exists(V_\circ, PK_\circ||M_\circ, sk_{1,b_1}, \ldots, sk_{n,b_n}, pad, b_1 \cdots b_n) \in \mathbb{T}_{\mathcal{E}^{-1}}$,
    then return $(sk_{1,b_1}, \ldots, sk_{n,b_n}, pad)$.
if $\mathbf{oVerify}(PK_\circ||M_\circ, V_\circ) = 0$,   //for invalid signatures, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \neq (SK'_\circ, \boldsymbol{sk}', \boldsymbol{pk}', PK_\circ) \in \mathbb{T}_{\mathcal{H}_\mathrm{sk}}$,  //for this bad event, $\mathcal{S}_5$ responds with random strings
    then $\widehat{pad} \twoheadleftarrow \{0,1\}^t$;
        for $i \in [1, n]$
            $\widehat{sk}_i \twoheadleftarrow \{0,1\}^{2\lambda}$;
        $\mathbb{T}_\mathcal{E} \leftarrow \mathbb{T}_\mathcal{E} \cup \{V_\circ, PK_\circ||M_\circ, \widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad}\}$;
        return $(\widehat{sk}_1, \ldots, \widehat{sk}_n, \widehat{pad})$.
$\widehat{\boldsymbol{sk}} \leftarrow \diamond$
if $\exists(PK_\circ, M_\circ, b_1^* \cdots b_n^*) \in \mathbb{T}_{\mathcal{H}_\mathrm{position}}$,
    then $\widehat{b}_1, \cdots \widehat{b}_n \leftarrow b_1^* \cdots b_n^*$;
    else $\widehat{b}_1, \cdots \widehat{b}_n \twoheadleftarrow \{0,1\}^n$; $\mathbb{T}_{\mathcal{H}_\mathrm{position}} \leftarrow \mathbb{T}_{\mathcal{H}_\mathrm{position}} \cup \{(PK_\circ, M_\circ, \widehat{b}_1, \cdots \widehat{b}_n)\}$;
if $\exists(SK_\circ, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_\mathrm{sk}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\exists(\diamond, \boldsymbol{sk}, \boldsymbol{pk}, PK_\circ) \in \mathbb{T}_{\mathcal{H}_\mathrm{sk}}$,
    then $\widehat{\boldsymbol{sk}} \leftarrow \boldsymbol{sk}$; $\widehat{\boldsymbol{pk}} \leftarrow \boldsymbol{pk}$;
if $\widehat{\boldsymbol{sk}} = \diamond$,
    then $\widehat{\boldsymbol{sk}} \twoheadleftarrow \{0,1\}^{4n\lambda}$;
    parse $\widehat{\boldsymbol{sk}}$ into $\big(\langle\widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle\widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\big)$;
    for $i \in [1, n]$
        $\widehat{pk}_{i,0} \leftarrow \mathcal{S}_9^{\mathcal{H}_\mathrm{OneWay}}(\widehat{sk}_{i,0})$, $\widehat{pk}_{i,1} \leftarrow \mathcal{S}_9^{\mathcal{H}_\mathrm{OneWay}}(\widehat{sk}_{i,1})$,
        $\mathbb{T}_{\mathcal{H}_\mathrm{OneWay}} \leftarrow \mathbb{T}_{\mathcal{H}_\mathrm{OneWay}} \cup \{(\widehat{sk}_{i,0}, \widehat{pk}_{i,0})\} \cup \{(\widehat{sk}_{i,1}, \widehat{pk}_{i,1})\}$;
    $\widehat{\boldsymbol{pk}} \leftarrow \big(\langle\widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle\widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\big)$;
    $\mathbb{T}_{\mathcal{H}_\mathrm{sk}} \leftarrow \mathbb{T}_{\mathcal{H}_\mathrm{sk}} \cup \{(\diamond, \widehat{\boldsymbol{sk}}, \widehat{\boldsymbol{pk}}, PK_\circ)\}$;
parse $\widehat{\boldsymbol{sk}}$ into $\big(\langle\widehat{sk}_{1,0}, \widehat{sk}_{1,1}\rangle, \ldots, \langle\widehat{sk}_{n,0}, \widehat{sk}_{n,1}\rangle\big)$;
parse $\widehat{\boldsymbol{pk}}$ into $\big(\langle\widehat{pk}_{1,0}, \widehat{pk}_{1,1}\rangle, \ldots, \langle\widehat{pk}_{n,0}, \widehat{pk}_{n,1}\rangle\big)$;
$\widehat{pad} \leftarrow 0 \cdots 0$;
$\mathbb{T}_{\mathcal{E}^{-1}} \leftarrow \mathbb{T}_{\mathcal{E}^{-1}} \cup \{(V_\circ, PK_\circ||M_\circ, \widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad}, \widehat{b}_1 \cdots, \widehat{b}_n)\}$;
return $(\widehat{sk}_{1,\widehat{b}_1}, \ldots, \widehat{sk}_{n,\widehat{b}_n}, \widehat{pad})$.
```

It's trivial that $\Pr[\mathcal{G}_{\mathcal{S}_9}^{\mathcal{F}_9,\mathcal{D}} = 1] = \Pr[\mathbf{Ideal}_\mathcal{S}^{\mathbf{OSIG},\mathcal{D}}]$, and thus we complete the entire proof.

B Ideal Objects

Random Oracle Model (ROM). Random oracle model is an idealized model proposed by Bellare and Rogaway [BR93]. ROM formalizes a model (a theoretical black box) which responds to any unique query with a truly random string, and if the query is repeated, the response would be consistent. More concretely, a random oracle model has a publicly accessible hash function $H : \{0,1\}^* \to \{0,1\}^n$ such that:

1. for any x, every bit of $H(x)$ is truly random;
2. for any $x \neq y$, $H(x)$ and $H(y)$ are independent.

Ideal Cipher Model (ICM). The ideal cipher model is another idealized model which is firstly proposed by Shannon [Sha49] and then formalized by Black [Bla06]. This model also responds to any unique query with a truly random string. While, instead of having a publicly accessible random function, ideal cipher model has a publicly accessible ideal cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Specifically, E is an ideal cipher along with a k-bit key and n-bit input/output such that:

1. for any pair (k, x), every bit of $E(k, x)$ is truly random;
2. for any fix key k, $E(k, *)$ is a random permutation;
3. for any $k_1 \neq k_2$ and (x, y), $E(k_1, x)$ and $E(k_2, y)$ are independent.

Moreover, any adversary interacting with an ideal cipher model would be given access to both the cipher and its inverse.