On the Relationship between Public Key Primitives via Indifferentiability

Shuang Hu[1,3], Bingsheng Zhang[1,2], Cong Zhang[1,2], and Kui Ren[1,2]*

[1] The State Key Laboratory of Blockchain and Data Security, Zhejiang University,
[2] Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, Hangzhou, China
[3] Zhejiang A&F University
{shuanghu, bingsheng, congresearch, kuiren}@zju.edu.cn

Recently, Masny and Rindal [MR19] formalized a notion called Endemic Oblivious Transfer (EOT), and they proposed a generic transformation from Non-Interactive Key Exchange (NIKE) to EOT with standalone security in the random oracle (RO) model. However, from the model level, the relationship between idealized NIKE and idealized EOT and the relationship between idealized elementary public key primitives have been rarely researched.

In this work, we investigate the relationship between ideal NIKE and ideal one-round EOT, as well as the relationship between ideal public key encryption (PKE) and ideal two-round Oblivious Transfer (OT), in the indifferentiability framework proposed by Maurer *et al.*(MRH04). Our results are threefold: Firstly, we model ideal PKE without public key validity test, ideal one-round EOT and ideal two-round OT in the indifferentiability framework. Secondly, we show that ideal NIKE and ideal one-round EOT are equivalent, and ideal PKE without public key validity test are equivalent to ideal two-round OT. Thirdly, we show a separation between ideal two-round OT and ideal one-round EOT, which implies a separation between ideal PKE and ideal NIKE.

Keywords: Indifferentiability, Idealized Model, NIKE, Endemic OT, PKE, OT

1 Introduction

Oblivious Transfer (OT) is one of the most important fundamental cryptographic primitives in secure Multi-Party Computation (MPC). Many well-known MPC protocols, such as Yao's GC [23], GMW [15], IPS [17], use OT as a key building block in their design. In practice, to achieve better online efficiency, MPC parties often prepare sufficiently many Random OT (ROT) instances in the offline phase, and then convert them to the needed OT instances accordingly in the online phase. The first non-interactive ROT protocol was proposed by Bellare and Micali [3]; the protocol can be completed within one-round, considering

* This paper is supported by the National Key R&D Program of China (No. 2021YFB3101601, No.2023YFB3106000) and the National 42 Natural Science Foundation of China (Grant No. 62072401 43 and No. 62232002).

simultaneous messaging, i.e., both parties can send messages to each other simultaneously in the same round. It is semi-honest secure under the DDH assumption in the Common Reference String (CRS) model. Later, Garg and Srinivasan [13] show that the ROT [3] can be upgraded to achieve malicious security using Groth-Sahai proof [16]. However, when considering malicious adversaries, due to its non-interactivity, the malicious sender can bias its output (m_0, m_1), and the malicious receiver can bias its output m_b, where $b \in \{0, 1\}$ is the receiver's choice bit. This property is later captured by the notion of Endemic Oblivious Transfer (EOT) introduced by Masny and Rindal [19]. The functionality of EOT is the same as ROT but it offers weaker security guarantees – the malicious party can fix its output arbitrarily. This type of weak ROT is also considered by Garg *et al.* [12], and the authors propose several one-round UC-secure EOT constructions under various assumptions in the CRS model. Recently, Zhou *et al.* [25] proposed many one-round UC/GUC-secure EOT in the (global) RO model.

In terms of the relationship between EOT and non-interactive key exchange (NIKE), Masny and Rindal [19] proposed a generic transformation from a key exchange protocol to an EOT protocol with standalone security in the RO model. Although the authors claim that if the key exchange protocol is one-round then the EOT can be completed within the same round, no security proof is provided. In terms of the relationship between OT and public key encryption (PKE), there are several related works in the standard model. Gertner *et al.* [14] showed that PKE with oblivious sampleable public key implies two-round OT, and PKE with oblivious sampleable ciphertext implies three round OT, but no formal proof was provided. Peikert *et al.* [21] proved that dual mode encryption implies OT. Friolo *et al.* [11] showed that two types of strong uniform PKE regarding public key and ciphertext imply strong uniform semi-honestly secure OT. Li *et al.* [18] showed that rerandomizable PKE implies OT.

However, nearly no research focused on the relationship between ideal model EOT and ideal model NIKE, or the relationship between ideal model OT and ideal model PKE, in the indifferentiability framework [20]. The relationship between elementary public key primitives under the ideal model is an interesting question that motivates our work.

The indifferentiability framework proposed by Maurer, Renner and Holenstein (MRH) [20] formalizes a set of necessary and sufficient conditions for one cryptosystem to securely be replaced with another one in an arbitrary environment. A number of cryptographic primitives justify the structural soundness via this framework; those primitives include hash functions [6, 9], blockciphers [1, 8, 10], domain extenders [7], authenticated encryption with associated data [2], and public key cryptosystems [24].

Compared to standard model, the indifferentiability framework considers cryptographic primitives in a more abstract level, which offers a measure for cryptographic ideal models and promotes a deeper understanding of the relationship between cryptographic primitives.

Relationship with Universal Composability (UC) [24]. Both the indifferentiability framework and the UC [5] framework define an "ideal" object for a given cryptographic concept, and they both use a simulation paradigm to define security and composition. However, the two notions have fundamental difference. In the UC framework, an ideal functionality specifies how a trusted third party solves a given cryptographic task abstractly, for example, an ideal functionality for public key encryption specifies how messages are passed. In contrast, in the indifferentiability framework, an ideal model specifies how to such a task concretely, in particular, the concrete algorithms along with the inputs and outputs interfaces for the ideal model are provided. In addition, discussing the relationship of two idealized models is natural in the indifferentiability framework, but it is unnatural in UC, since UC mainly measures how close a standard model construction (maybe with the help of some ideal functionality) is to a desired ideal functionality. Moreover, it is much more convenient to model ideal constant-round primitives, such as NIKE, PKE and two-round OT, in the indifferentiability framework than in the UC framework.

Hereby, we ask the following question in the indifferentiability framework:

What is the relationship between ideal NIKE and ideal one-round EOT?
Moreover, what is the relationship between ideal PKE and ideal two-round OT?

1.1 Our Results

In this work, we investigate the above question thoroughly. Our contribution can be summarized as follows.

Modeling Ideal one-round EOT and Ideal two-round OT. Following the similar methodology in Zhandry and Zhang [24], we model ideal one-round EOT protocol as four ideal algorithms $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$[4]. $\mathsf{I.EOT}_{\mathsf{A}_1}$ and $\mathsf{I.EOT}_{\mathsf{B}_1}$ can be run independently and simultaneously by two parties with their secret inputs. First, a sender runs $\mathrm{PK}_1 \leftarrow \mathsf{I.EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$ and a receiver runs $Q \leftarrow \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)$, and they send PK_1, Q to each other, respectively. Next, the sender runs $(K_0, K_1) \leftarrow \mathsf{I.EOT}_{\mathsf{B}_2}(\mathrm{SK}_1, Q)$ and the receiver runs $K_b \leftarrow \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, b, \mathrm{PK}_1)$ locally, such that K_b equals one of K_0, K_1. Note that the communication can be done in a simultaneous round, so we call this protocol ideal "one-round" EOT. In addition, we model ideal two-round OT protocol as three ideal algorithms $(\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3)$, where the three algorithms should be run sequentially.First, a receiver runs $Q \leftarrow \mathsf{I.OT}_1(\mathrm{SK}_0, b)$ and sends Q to a sender.Next, the sender runs $w \leftarrow \mathsf{I.OT}_2(Q, \mathrm{SK}_1, m_0, m_1)$ and sends w to the receiver. Finally, the receiver runs $m_b \leftarrow \mathsf{I.OT}_3(\mathrm{SK}_0, b, w)$ and output m_b that should equal m_0 or m_1. Here the communication should be finished in two rounds, so we call this protocol ideal "two-round" OT.

Equivalence of Ideal one-round EOT and Ideal NIKE. We show the equivalence between ideal NIKE and ideal one-round EOT in the indifferentiability framework. Concretely, we provide transformations from ideal NIKE to

[4] We use the prefix "I." to denote ideal schemes, protocols and algorithms, and use the prefix "Π." to denote our constructed schemes, protocols and algorithms.

ideal EOT and back and we prove the security of the transformations in the indifferentiability framework.

Equivalence of Ideal two-round OT and Ideal PKE without Public Key Validity Test. We show the equivalence of ideal PKE without public key validity test and ideal two-round OT In the indifferentiability framework. In particular, we provide transformations from ideal PKE to ideal two-round OT and back, and we prove the security of the transformations in both directions in the indifferentiability framework.

Separation between Ideal two-round OT and Ideal one-round EOT. We show a separation between ideal two-round OT and ideal one-round EOT in the following steps: firstly, we show that ideal one-round EOT can be used to construct ideal two-round OT, secondly, we show the inverse is not true, thus the two ideal models are not equivalent. This also implies a separation between ideal NIKE and ideal PKE, according to the two equivalence relations above-mentioned.

Our results can be depicted in Figure 1, where the one-way arrows indicate that the ideal models at the end of the arrow can be transformed into the ideal models at the top of the arrow, and a slash in the middle of an arrow indicate that the ideal model at the end of the arrow are separated from the ideal models at the top of the arrow, namely, the former cannot be transformed into the latter. Besides, the transformations' security and the separations are proved in the theorems above the arrows.

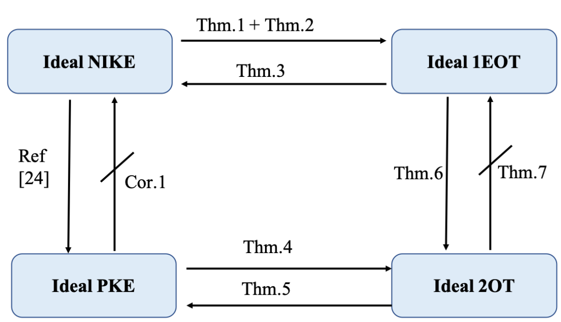

Fig. 1: Roadmap of our results

1.2 Our Techniques

In this section, we describe our techniques for proving the two equivalence results. To prove the equivalence of ideal EOT and ideal NIKE, we observe that an ideal NIKE has two algorithms: key generation algorithm I.symKG and shared key algorithm $\mathsf{I.symSHK}$, with the correctness requirement $\mathsf{I.symSHK}(\mathrm{SK}_0, \mathsf{I.symKG}(\mathrm{SK}_1)) = \mathsf{I.symSHK}(\mathrm{SK}_1, \mathsf{I.symKG}(\mathrm{SK}_0))$. It can be observed that the algorithms of an ideal NIKE have a certain "symmetry", so we also call it "ideal symmetric NIKE". However, an ideal EOT has four algorithms with the outputs of two parties only partially identical. To make the proof easier, we introduce a new primitive called ideal asymmetric NIKE, which has four algorithms and the two parties can obtain the same output. To prove equivalence of ideal EOT and ideal NIKE, firstly, we construct an ideal asymmetric NIKE from an ideal NIKE; secondly, we construct an ideal EOT from an ideal asymmetric NIKE; finally, we construct an ideal NIKE from an ideal EOT. And we prove the security of our constructions. To prove the equivalence of ideal PKE and ideal two-round OT is much easier.

Below, we describe our detailed techniques.

Construct Ideal Asymmetric NIKE from Ideal NIKE.
An ideal asymmetric NIKE has four interfaces ($\mathsf{I.asyKG}_0$, $\mathsf{I.asyKG}_1$, $\mathsf{I.asySHK}_0$, $\mathsf{I.asySHK}_1$), where $\mathsf{I.asyKG}_0$, $\mathsf{I.asyKG}_1$ are key generation algorithms, and $\mathsf{I.asySHK}_0$,$\mathsf{I.asySHK}_1$ are shared key algorithms. These algorithms can be divided into "the left part" (consisting of ($\mathsf{I.asyKG}_0$,$\mathsf{I.asySHK}_0$) and their inputs and outputs) and "the right part" (consisting of ($\mathsf{I.asyKG}_1$,$\mathsf{I.asySHK}_1$ and their inputs and outputs). The correctness requires that $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathsf{I.asyKG}_1(\mathrm{SK}_1))$=$\mathsf{I.asySHK}_1(\mathrm{SK}_1, \mathsf{I.asyKG}_0(\mathrm{SK}_0))$. Note that the inputs for $\mathsf{I.asySHK}_0$ should have SK_0 from "the left part" and $\mathsf{I.asyKG}_1(\mathrm{SK}_1)$ from "the right part", and the inputs for $\mathsf{I.asySHK}_1$ should have SK_1 from "the right part" and $\mathsf{I.asyKG}_0(\mathrm{SK}_0))$ from "the left part".

Our goal is to combine an ideal NIKE and idealized models such as RO model [4] and ideal cipher model [22], to construct an ideal asymmetric NIKE. Our construction techniques are described as follows, where we use two random oracles H_0, H_1 and two ideal permutations [24] P_0, P_1. Simply setting $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1$ to be $\mathsf{I.symKG}$ is not enough, since we need two random oracles to process the inputs of $\Pi.\mathsf{asyKG}_0$ and $\Pi.\mathsf{asyKG}_1$, thus allowing $\mathsf{I.symKG}$ to distinguish the two inputs. Besides, the two algorithms $\Pi.\mathsf{asySHK}_0$ and $\Pi.\mathsf{asySHK}_1$ need to distinguish "the left part" public key PK_0 output by $\Pi.\mathsf{asyKG}_0$ from "the right part" public key PK_1 output by $\Pi.\mathsf{asyKG}_1$, so we use two ideal permutations P_0, P_1. P_0, P_1 enable converting $\mathrm{PK}_0, \mathrm{PK}_1$ back to the outputs of $\mathsf{I.symKG}$, thus serving as valid inputs for $\mathsf{I.symSHK}$.

Following the techniques, we construct an ideal asymmetric NIKE as follows: $\Pi.\mathsf{asyKG}_0 = P_0(\mathsf{I.symKG}(H_0(\mathrm{SK})))$, $\Pi.\mathsf{asyKG}_1$=$P_1(\mathsf{I.symKG}(H_1(\mathrm{SK})))$, $\Pi.\mathsf{asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1) = \mathsf{I.symSHK}(H_0(\mathrm{SK}_0), P_1^{-1}(\mathrm{PK}_1))$,and $\Pi.\mathsf{asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0) = \mathsf{I.symSHK}(H_1(\mathrm{SK}_1), P_0^{-1}(\mathrm{PK}_0))$.

This construction is indifferentiable from an ideal asymmetric NIKE. To prove this, we employ special/careful simulation strategies for the adversarial interfaces $H_0, H_1, P_0, P_1, P_0^{-1}, P_1^{-1}$, $\mathsf{I.symKG}$, $\mathsf{I.symSHK}$, via a sequence of hybrid games, where each game corresponds to one type of differentiation attack. By showing that these attacks do not work in each game, we prove the indistinguishability of our construction and ideal asymmetric NIKE scheme in the indifferentiability framework, and we formalize it in Theorem 1 and provide a proof.

Construct Ideal One-Round EOT from Ideal Asymmetric NIKE.
Our goal is to combine an ideal asymmetric NIKE and idealized objects like random oracles to construct an ideal model one-round EOT $\Pi.\mathsf{EOT}$ consisting of four algorithms $(\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$ using an ideal asymmetric NIKE consisting of four algorithms $(\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$. Since the algorithm interfaces of an ideal EOT and an ideal asymmetric NIKE are mostly similar, a natural idea is to correspond an ideal asymmetric NIKE's four algorithms one-to-one to the constructed ideal EOT's four algorithms, with appropriate adjustments. Our construction follows the steps below, in which we use four random oracles H_0, H_1, H_2, H_3 and an ideal cipher [24] $(\mathcal{E}, \mathcal{E}^{-1})$, where $\mathcal{E}$ is an encryption algorithm and $\mathcal{E}^{-1}$ is a decryption algorithm.

By definition, $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ should take $\Pi.\mathsf{EOT}_{\mathsf{A}_1}$'s output Q and a secret string SK_1 as the input and output two strings K_0, K_1. A natural idea is to split Q as two parts to somehow correspond to K_0, K_1. A first attempt is to construct $Q \leftarrow \Pi.\mathsf{EOT}_{\mathsf{A}_1}(b, \mathrm{SK}_0)$ as this: $Q_{1-b} = H_2(\mathrm{SK}_0, b)$, $Q_b = \mathsf{I.asyKG}_0(\mathrm{SK}_0) \oplus H_3(Q_{1-b})$, $Q = Q_0 \| Q_1$. Following this construction, it holds that $\mathsf{I.asyKG}_0(\mathrm{SK}_0) = \mathrm{PK}_0 = H_3(Q_{1-b}) \oplus Q_b$ for both $b = 0$ and $b = 1$. Namely, for $(Q_0, Q_1) \leftarrow \Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, 0)$ and $(Q'_0, Q'_1) \leftarrow \Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, 1)$, a fixed relation $H_3(Q_1) \oplus Q_0 = H_3(Q'_0) \oplus Q'_1$ holds. However, for an ideal algorithm $\mathsf{I.EOT}_{\mathsf{A}_1}$, this fixed relation should not hold, otherwise the randomness of $\mathsf{I.EOT}_{\mathsf{A}_1}$ is violated. To remove the fixed relation, we embed $\mathsf{I.asyKG}_0(\mathrm{SK}_0)$ in an ideal cipher $\mathcal{E}$ using $H_3(Q_{1-b})$ as the key. We define $Q \leftarrow \Pi.\mathsf{EOT}_{\mathsf{A}_1}(b, \mathrm{SK}_0)$ as this: $Q_{1-b} = H_2(\mathrm{SK}_0, b)$, $Q_b = \mathcal{E}(H_3(Q_{1-b}), \mathsf{I.asyKG}_0(\mathrm{SK}_0))$, output $Q = Q_0 \| Q_1$.

Following the techniques, we define $\mathrm{PK}_1 \leftarrow \Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1) = \mathsf{I.asyKG}_1(\mathrm{SK}_1)$, $K_b \leftarrow \Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{PK}_1, \mathrm{SK}_0, b) = H_b(\mathsf{I.asySHK}_0(\mathrm{PK}_1, \mathrm{SK}_0))$, where $H_b = H_0$ when $b = 0$ and $H_b = H_1$ when $b = 1$. And we define $(K_0, K_1) \leftarrow \Pi.\mathsf{EOT}_{\mathsf{B}_2}(Q, \mathrm{SK}_1)$ as follows: $Q = Q_0 \| Q_1$, $A_0 = \mathcal{E}^{-1}(H_3(Q_1), Q_0)$, $A_1 = \mathcal{E}^{-1}(H_3(Q_0), Q_1)$, $K_0 \leftarrow H_0(\mathsf{I.asySHK}_1(A_0, \mathrm{SK}_1))$, $K_1 \leftarrow H_1(\mathsf{I.asySHK}_1(A_1, \mathrm{SK}_1))$.

The joint construction of $(\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$ are indifferentiable from an ideal EOT $\mathsf{I.EOT}$, which we formalize in Theorem 2 and provide a proof.

Construct Ideal NIKE from Ideal One-Round EOT.
Our goal is to combine ideal EOT protocol and random oracles to obtain an ideal model symmetric NIKE consisting of two algorithms $(\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$. Our construction follows the steps below, in which we use two random oracles H_0, H_1.

We need to use the four algorithms of an ideal EOT to construct two algorithms of an ideal NIKE, a natural idea is to use the combined outputs of $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}$ as the output of $\Pi.\mathsf{symKG}$, and to use the combined outputs of $\mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_2}$ as the output of $\Pi.\mathsf{symKG}$. Wlog, the inputs of $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}$ should be different, thus we use a random oracle H_1 to process an input SK_0 of $\Pi.\mathsf{symKG}$ to produce another input $\mathrm{SK}_B = H_1(\mathrm{SK}_0)$. Then we let $\mathrm{SK}_0, \mathrm{SK}_B$ be the inputs of $\mathsf{I.EOT}_{\mathsf{A}_1}$ and $\mathsf{I.EOT}_{\mathsf{B}_1}$, respectively.

To make the outputs of adversarial interface and honest interface consistent, we carefully provide all necessary information in the adversarial interfaces. First, we define $\Pi.\mathsf{symKG}(\mathrm{SK}_0) = (\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, 0), \mathsf{I.EOT}_{\mathsf{B}_1}(H_1(\mathrm{SK}_0)))$. Next, we define $\Pi.\mathsf{symSHK} = H_0(\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1^B, 0), \mathsf{LoR}_0(\mathsf{I.EOT}_{\mathsf{B}_2}(H_1(\mathrm{SK}_0),\ \mathrm{PK}_1^A))$.

This joint construction of ($\Pi.\mathsf{symKG}$, $\Pi.\mathsf{symSHK}$) is indifferentiable from an ideal symmetric NIKE $\mathsf{I.symNIKE}$ in Def. 2, which we formalize in Theorem 3 and provide a proof.

Construct Ideal Two-Round OT from Ideal PKE.
Our goal is to combine an ideal PKE and random oracles to construct an ideal two-round OT $\Pi.\mathsf{2OT}$, consisting of three algorithms $(\Pi.\mathsf{OT}_1, \Pi.\mathsf{OT}_2, \Pi.\mathsf{OT}_3)$. Our construction follows the steps below, in which we use four random oracles H_0, H_1, H_2, H_3, a random permutation P_0 and an ideal cipher $\mathcal{E}$.

We first attempt to define $Q \leftarrow \Pi.\mathsf{OT}_1(b, \mathrm{SK}) = \mathsf{I.KGEN}(\mathrm{SK}) \oplus H_2(\mathbf{b}^{\ell_0})$, where $\ell_0 = |\mathrm{PK}|$. However, this construction implies a fixed relation $\Pi.\mathsf{OT}_1(\mathrm{SK}, 0) \oplus \Pi.\mathsf{OT}_1(\mathrm{SK}, 1) = H_2(\mathbf{0}^{\ell_0}) \oplus H_2(\mathbf{1}^{\ell_0})$, and the relation should not hold for an ideal algorithm $\mathsf{I.OT}_1$. Therefore, we incorporate b in a random oracle and embed $\mathsf{KGEN}(\mathrm{SK})$ in an ideal cipher to remove the fixed relation, using the similar strategy in the construction of an ideal EOT from an ideal asymmetric NIKE. To make the outputs of adversarial interface and honest interface consistent, we carefully provide all necessary information in the adversarial interfaces.

Following these techniques, firstly, we define $Q \leftarrow \Pi.\mathsf{OT}_1(\mathrm{SK}, b)$ as this: $\mathrm{PK} = \mathsf{KGEN}(\mathrm{SK})$, $Q_{1-b} = H_2(\mathrm{SK}, b)$, $Q_b = \mathcal{E}(H_3(Q_{1-b}), \mathrm{PK})$, output $Q = Q_0 \| Q_1$. Secondly, we define $w \leftarrow \Pi.\mathsf{OT}_2(Q, m_0, m_1, \overline{\mathrm{SK}})$ as this: $\mathrm{PK}_0 := \mathcal{E}^{-1}(H_3(Q_1), Q_0)$, $\mathrm{PK}_1 := \mathcal{E}^{-1}(H_3(Q_0), Q_1)$, $C_0 \leftarrow \mathsf{ENC}(\mathrm{PK}_0, m_0, H_0(\overline{\mathrm{SK}}, m_1))$, $C_1 \leftarrow \mathsf{ENC}(\mathrm{PK}_1, m_1, H_1(\overline{\mathrm{SK}}, m_0))$, output $w := P_0(C_0 \| C_1)$. Finally, we define $m_b \leftarrow \Pi.\mathsf{OT}_3(w, \mathrm{SK}, b)$ as follows: $C_0 \| C_1 = P_0^{-1}(w)$, $m_b = \mathsf{DEC}(\mathrm{SK}, C_b)$.

This joint construction $\Pi.\mathsf{2OT}$=($\Pi.\mathsf{OT}_1$, $\Pi.\mathsf{OT}_2$, $\Pi.\mathsf{OT}_3$) is indifferentiable from ideal two-round OT $\mathsf{I.2OT}$, and we formalize it in Theorem 5 and provide a proof.

Construct Ideal PKE from Ideal Two-Round OT.
Our goal is to combine ideal two-round OT protocol and random oracles to construct an indifferentiable PKE $\Pi.\mathsf{PKE}$ consisting of three algorithms $(\Pi.\mathsf{KGEN}, \Pi.\mathsf{ENC}, \Pi.\mathsf{DEC})$. A natural idea is to set the choice bit b in the OT protocol as 0, and let the two messages from the sender be the same, and use the sender's secret input $\overline{\mathrm{SK}}$ as the randomness for the encryption. Following this idea, we construct $\Pi.\mathsf{PKE}$ as following: $\Pi.\mathsf{KGEN}(\mathrm{SK}) = \mathsf{I.2OT}_1(0, \mathrm{SK}) = \mathrm{PK}$, $\Pi.\mathsf{ENC}(\mathrm{PK}, m, \mathrm{nonce}) = \mathsf{LoR}_0(\mathsf{I.2OT}_2(m, m, \mathrm{PK}, \mathrm{nonce})) = C$, where $\mathrm{nonce} = \overline{\mathrm{SK}}$, and $\Pi.\mathsf{DEC}(\mathrm{SK}, C) = \mathsf{I.2OT}_3(C, \mathrm{SK}, 0) = m$.

This joint construction $\Pi.\mathsf{PKE}$=($\Pi.\mathsf{KGEN}$, $\Pi.\mathsf{ENC}$, $\Pi.\mathsf{DEC}$) is indifferentiable from an ideal PKE $\mathsf{I.PKE}$, and we show it in Theorem 4.

2 Preliminaries

Notations. Let $\lambda \in \mathbb{N}$ be the security parameter for all the definitions in this paper, and let "PPT" denote probabilistic polynomial time. Let $y \leftarrow \mathsf{f}(x)$ denote running a probabilistic algorithm f with an input x and obtaining an output y. Denote by $\mathsf{negl}(\lambda)$ a negligible function of λ. Let $\mathbf{1}^n$ and $\mathbf{0}^n$ denote a n-bit string with every bit being 1 and a n-bit string with every bit being 0, respectively. Let $a\|b$ denote concatenating a string a and b from left-to-right order, and $\mathsf{LoR_b}(x)$ denote a function that outputs the left half of x if $b = 0$, or the right half of x if $b = 1$.

2.1 Indifferentiability Framework

When it comes to the framework of indifferentiability, we typically consider that a cryptosystem implements either some ideal objects $\mathcal{F}$, or a construction $C^{\mathcal{F}'}$ which applies underlying ideal objects $\mathcal{F}'$.

Definition 1. *[**Indifferentiability** [20]] Let Σ_1 and Σ_2 be two cryptosystems and $\mathcal{S}$ be a simulator. The indifferentiability advantage of a differentiator $\mathcal{D}$ against (Σ_1, Σ_2) with respect to $\mathcal{S}$ is*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}(1^\lambda) := \Pr[\mathrm{Real}_{\Sigma_1,\mathcal{D}}(1^\lambda)] - \Pr[\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}(1^\lambda)],$$

where the real game $\mathrm{Real}_{\Sigma_1,\mathcal{D}}$ and the ideal game $\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}$ are defined in Figure 2. We say Σ_1 is indifferentiable from Σ_2, if there exists an efficient simulator $\mathcal{S}$ such that for any probabilistic polynomial time differentiator $\mathcal{D}$, the advantage above is negligible. Moreover, we say Σ_1 is statistically indifferentiable from Σ_2, if there exists a PPT simulator such that, for any unbounded differentiator $\mathcal{D}$, the advantage above is negligible.

$\mathrm{Real}_{\Sigma_1,\mathcal{D}}(1^\lambda)$:	HonestR(X)	$\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}(1^\lambda)$:	HonestI(X)
$b \leftarrow \mathcal{D}^{\mathrm{HonestR,AdvR}}$	Return $\Sigma_1.\mathrm{hon}(X)$.	$b \leftarrow \mathcal{D}^{\mathrm{HonestI,AdvI}}$	Return $\Sigma_2.\mathrm{hon}(X)$.
Return b.	AdvR(X)	Return b.	AdvI(X)
	Return $\Sigma_1.\mathrm{adv}(X)$.		Return $\mathsf{Sim}^{\Sigma_2.\mathrm{adv}(\cdot)}(X)$.

Fig. 2: Indifferentiability of Σ_1 and Σ_2

Below, we also use the notations in [2] and consider the definition above to two systems with interfaces as:

$$(\Sigma_1.\mathrm{hon}(\mathrm{X}), \Sigma_1.\mathrm{adv}(\mathrm{x})) := (\mathrm{C}^{\mathcal{F}_1}(\mathrm{X}), \mathcal{F}_1(\mathrm{x})); (\Sigma_2.\mathrm{hon}(\mathrm{X}), \Sigma_2.\mathrm{adv}(\mathrm{x})) := (\mathcal{F}_2(\mathrm{X}), \mathcal{F}_2(\mathrm{x})),$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two ideal objects sampled from their distributions and $\mathrm{C}^{\mathcal{F}_1}$ is a construction of $\mathcal{F}_2$ by calling $\mathcal{F}_1$.

Next, we provide the definition of ideal NIKE without public key validity test[5], which means that no PPT algorithm can distinguish a uniformly sampled string in the public key space from a key output by the key generation algorithm, referring to [24].

Definition 2. *[**Ideal NIKE** [24]] Let $\mathcal{X}, \mathcal{Y}, \mathcal{W} \in \omega(\log \lambda)$ be three sets. We denote $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$ as the set of all injections that map $\mathcal{X}$ to $\mathcal{Y}$ and $\mathcal{G}[\mathcal{X} \times \mathcal{Y} \to \mathcal{K}]$ as the set of functions that map $\mathcal{X} \times \mathcal{Y}$ to $\mathcal{K}$.*

We define an ideal NIKE scheme $\mathsf{I.symNIKE} = (\mathsf{I.symGEN}, \mathsf{I.symSHK})$ as the set of all function pairs (f, g) such that: (1) $f \in \mathcal{F}$, $g \in \mathcal{G}$; (2) $\forall x_1, x_2 \in \mathcal{X}$, $g(x_1, f(x_2)) = g(x_2, f(x_1))$; (3) $g(x_1, y_1) = g(x_2, y_2) \Rightarrow (x_1 = x_2 \wedge y_1 = y_2) \vee (y_1 = f(x_2) \wedge y_2 = f(x_1))$.

The definition of ideal PKE without public key validity test[6] is provided below, which means that no PPT algorithm can distinguish whether a given string is uniformly sampled from the public key space or generated by the key generation algorithm, referring to [24].

Definition 3. *[**Ideal PKE** [24]] Let $\mathcal{X}, \mathcal{Y}, \mathcal{M}, \mathcal{R}, \mathcal{C} \in \omega(\log \lambda)$ be five sets.We denote $\mathcal{F}_1[\mathcal{X} \to \mathcal{Y}]$ as the set of all injections that map $\mathcal{X}$ to $\mathcal{Y}$; $\mathcal{F}_2[\mathcal{Y} \times \mathcal{M} \times \mathcal{R} \to \mathcal{C}]$ as the set of all injections that map $\mathcal{Y} \times \mathcal{M} \times \mathcal{R}$ to $\mathcal{C}$ and $\mathcal{F}_3[\mathcal{C} \times \mathcal{X} \to \mathcal{M} \cup \bot]$ as the set of all functions that map $\mathcal{X} \times \mathcal{C}$ to $\mathcal{M} \cup \bot$.*

We define an ideal PKE scheme $\mathsf{I.PKE} = (\mathsf{I.KGEN}, \mathsf{I.ENC}, \mathsf{I.DEC})$ as the set of all function tuples (f_1, f_2, f_3) such that: (1) $f_1 \in \mathcal{F}_1$, $f_2 \in \mathcal{F}_2$ and $f_3 \in \mathcal{F}_3$; (2)$\forall x \in \mathcal{X}$, $m \in \mathcal{M}$ and $r \in \mathcal{R}$, $f_3(x, f_2(f_1(x), m, r)) = m$; (3)$\forall x \in \mathcal{X}, c \in \mathcal{C}$, if there is no $(m, r) \in \mathcal{M} \times \mathcal{R}$ such that $f_2(f_1(x), m, r) = c$, then $f_3(x, c) = \bot$.

3 EOT and NIKE are Equivalent in Indifferentiability Framework

3.1 Indifferentiable Asymmetric NIKE from Ideal Symmetric NIKE

First, we define ideal asymmetric NIKE and ideal one-round EOT.

Definition 4. *[**Ideal Asymmetric NIKE**] Let $\mathcal{X}_1, \mathcal{Y}_1, \mathcal{X}_2, \mathcal{Y}_2, \mathcal{K} \in \omega(\log \lambda)$ be five sets. Let $\mathcal{G}_1 := \{\mathcal{X}_1 \mapsto \mathcal{Y}_1\}$ be a family of injective functions that maps an element from $\mathcal{X}_1$ to an element in $\mathcal{Y}_1$.Let $\mathcal{G}_2 := \{\mathcal{X}_2 \mapsto \mathcal{Y}_2\}$ be a family of injective functions that maps an element from $\mathcal{X}_2$ to an element in $\mathcal{Y}_2$. Let $\mathcal{F}_1 := \{\mathcal{X}_2 \times \mathcal{Y}_1 \mapsto \mathcal{K}\}$ be a family of functions that maps an element from $\mathcal{X}_2 \times \mathcal{Y}_1$*

[5] In the following, we abbreviate ideal NIKE without public key validity test as ideal NIKE.

[6] In the following, we abbreviate ideal PKE without public key validity test as ideal PKE.

to an element in $\mathcal{K}$. Let $\mathcal{F}_2 := \{\mathcal{X}_1 \times \mathcal{Y}_2 \mapsto \mathcal{K}\}$ be a family of functions that maps an element from $\mathcal{X}_1 \times \mathcal{Y}_2$ to an element in $\mathcal{K}$.

We define an ideal asymmetric NIKE as a set of all function tuples (g_1, g_2, f_1, f_2) such that: (1) for $i \in [2]$, $g_i \in \mathcal{G}_i$ and $f_i \in \mathcal{F}_i$; (2) $\forall x_1 \in \mathcal{X}_1$, $\forall x_2 \in \mathcal{X}_2$, $f_1(g_2(x_2), x_1) = f_2(g_1(x_1), x_2)$.

Definition 5. *[**Ideal One-Round EOT**] Let $\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}_1, \mathcal{Y}_2, \mathcal{K} \in \omega(\log \lambda)$ be five sets. Let $\mathcal{F}_1 := \{\{0,1\} \times \mathcal{X}_1 \mapsto \mathcal{Y}_1\}$ be a family of injective functions that maps an element from $\{0,1\} \times \mathcal{X}_1$ to an element in $\mathcal{Y}_1$. Let $\mathcal{F}_2 := \{\mathcal{X}_2 \mapsto \mathcal{Y}_2\}$ be a family of functions that maps an element in $\mathcal{X}_2$ to an element in $\mathcal{Y}_2$. Let $\mathcal{F}_3 := \{\{0,1\} \times \mathcal{X}_1 \times \mathcal{Y}_2 \mapsto \mathcal{K}\}$ be a family of functions that maps an element in $\{0,1\} \times \mathcal{X}_1 \times \mathcal{Y}_2$ to an element in $\mathcal{K}$. Let $\mathcal{F}_4 := \{\mathcal{X}_2 \times \mathcal{Y}_1 \mapsto \mathcal{K} \times \mathcal{K}\}$ be a family of functions that maps an element in $\mathcal{X}_2 \times \mathcal{Y}_1$ to an element in $\mathcal{K} \times \mathcal{K}$.*

We define ideal one-round $\mathsf{IEOT} = (\mathsf{IEOT}_{\mathsf{A}_1}, \mathsf{IEOT}_{\mathsf{A}_2}, \mathsf{IEOT}_{\mathsf{B}_1}, \mathsf{IEOT}_{\mathsf{B}_2})$ as the set of all function tuples (f_1, f_2, f_3, f_4) such that: (1) for $i \in [4]$, $f_i \in \mathcal{F}_i$; (2)$\forall b \in \{0,1\}$, $\forall x_1 \in \mathcal{X}_1$,$\forall x_2 \in \mathcal{X}_2$: $f_4(f_1(b, x_1), x_2) = (y_0, y_1)$ and $f_3(b, x_1, f_2(x_2)) = y_b$.

Parameters. Denote the secret key space, public key space and shared key space of $\mathsf{I.symKG}$ as $\{0,1\}^{n_1(\lambda)}$, $\{0,1\}^{n_2(\lambda)}$, $\{0,1\}^{n_3(\lambda)}$, respectively. Denote the two secret key spaces, public key space and shared key space of $\mathsf{I.asyKG}$ as $\{0,1\}^{\ell_1(\lambda)}$, $\{0,1\}^{\ell_3(\lambda)}$, $\{0,1\}^{\ell_2(\lambda)}$, $\{0,1\}^{n_3(\lambda)}$, respectively. Denote the two secret key spaces, two public key spaces and key space of $\mathsf{I.EOT}$ as $\{0,1\}^{\ell_1(\lambda)}$, $\{0,1\}^{\ell_3(\lambda)}$, $\{0,1\}^{\ell_2(\lambda)}$, $\{0,1\}^{\ell_4(\lambda)}$, $\{0,1\}^{n_3(\lambda)}$, respectively.

Construction. The construction of an asymmetric NIKE scheme $\Pi.\mathsf{asyNIKE} = (\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1)$ (wlog, assuming that PK_1 and PK_0 for are of equal length) from an ideal symmetric NIKE scheme $\mathsf{I.symNIKE}$, is described below, where H_0, H_1 are random oracles, P_0, P_1 are random permutations, defined below: $H_0 : \{0,1\}^* \mapsto \{0,1\}^{n_1(\lambda)}$, $H_1 : \{0,1\}^* \mapsto \{0,1\}^{n_1(\lambda)}$, $P_0 : \{0,1\}^{\ell_2(\lambda)} \mapsto \{0,1\}^{\ell_2(\lambda)}$, $P_1 : \{0,1\}^{\ell_2(\lambda)} \mapsto \{0,1\}^{\ell_2(\lambda)}$.

- $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0) = P_0(\mathsf{I.symKG}(H_0(\mathrm{SK}_0)))$;
- $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1) = P_1(\mathsf{I.symKG}(H_1(\mathrm{SK}_1)))$;
- $\Pi.\mathsf{asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1) = \mathsf{I.symSHK}(H_0(\mathrm{SK}_0), P_1^{-1}(\mathrm{PK}_1))$;
- $\Pi.\mathsf{asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0) = \mathsf{I.symSHK}(H_1(\mathrm{SK}_1), P_0^{-1}(\mathrm{PK}_0))$.

Theorem 1. *The constructed scheme $\Pi.\mathsf{asyNIKE}$ in Sec. 3.1, with access to an ideal symmetric NIKE scheme $\mathsf{I.symNIKE}$, random oracles H_0, H_1 and random permutations P_0, P_1, is indifferentiable from an ideal asymmetric NIKE scheme as in Def. 4. More precisely, there exists a simulator $\mathcal{S}$ such that for all polynomial q-query distinguisher $\mathcal{D}$, the distinguishing advantage $\mathrm{Adv}^{\mathsf{indif}}_{\Pi.\mathsf{asyNIKE},\mathsf{I.asyNIKE},\mathcal{S},\mathcal{D}}(1^\lambda)$ satisfies the following:*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi.\mathsf{asyNIKE},\mathsf{I.asyNIKE},\mathcal{S},\mathcal{D}}(1^\lambda) \le \frac{2q^2}{2^{n_1(\lambda)}} + \frac{4q^2}{2^{n_2(\lambda)}} + \frac{3q^2}{2^{\ell_1(\lambda)}} + \frac{2q^2}{2^{\ell_3(\lambda)}} + \frac{q^2}{2^{\ell_2(\lambda)}} + \frac{2q^2}{2^{\ell_4(\lambda)}} \le \mathsf{negl}(\lambda),$$

Proof. By the definition of indifferentiability, in the real world, the differentiator $\mathcal{D}$ has oracle access to ($\Pi.\mathsf{asyKG}_0$, $\Pi.\mathsf{asyKG}_1$, $\Pi.\mathsf{asySHK}_0$, $\Pi.\mathsf{asySHK}_1$) via the honest interface and oracle access to ($H_0, H_1, \mathsf{I.symKG}$, $\mathsf{I.symSHK}$, P_0, P_0^{-1}, P_1, P_1^{-1}) via the adversarial interface. In the ideal world, the differentiator $\mathcal{D}$ has oracle access to ($\mathsf{I.asyKG}_0$, $\mathsf{I.asyKG}_1$, $\mathsf{I.asySHK}_0$, $\mathsf{I.asySHK}_1$) via the honest interface and access to $\mathcal{S}$ via the adversarial interface. Therefore, to establish a proof using the simulation paradigm, in the ideal world we need to build a PPT simulator $\mathcal{S}$ that simulates the oracles for the adversarial interface properly, by making queries to the oracles for the honest interface, such that for any PPT $\mathcal{D}$, the view in the real world is computationally close to the view in the ideal world.

Before describing the simulator, we first specify some parameters: 1) The differentiator $\mathcal{D}$ can make at most q queries to the oracles, where $q = \text{poly}(\lambda)$; 2) There are eight types of queries at the adversarial interface, corresponding to the eight oracles ($H_0, H_1, \mathsf{I.symKG}$, $\mathsf{I.symSHK}$, P_0, P_0^{-1}, P_1, P_1^{-1}); 3) In the real game, the queries at adversarial interface are responded by the corresponding oracles. And in the ideal game, the queries at adversarial interface are simulated by the simulator. The simulator maintains a table for each oracle at the adversarial interface, in the following forms, where the subscript indicates the corresponding oracle: $T_{\mathsf{H}_0} = (\text{SK}_0, \text{sk}_0, \text{pk}_0, \text{PK}_0)$, $T_{\mathsf{H}_1} = (\text{SK}_1, \text{sk}_1, \text{pk}_1, \text{PK}_1)$, $T_{\mathsf{P}_0} = (\text{SK}_0, \text{sk}_0, \text{pk}_0, \text{PK}_0)$, $T_{\mathsf{P}_0^{-1}} = (*, *, \text{pk}_0, \text{PK}_0)$, $T_{\mathsf{P}_1} = (\text{SK}_1, \text{sk}_1, \text{pk}_1, \text{PK}_1)$, $T_{\mathsf{P}_1^{-1}} = (*, *, \text{pk}_1, \text{PK}_1)$, $T_{\mathsf{symKG}} = (*, \text{sk}, \text{pk}, *)$, $T_{\mathsf{symSHK}} = (\widehat{\text{sk}}, \tilde{\text{pk}}, K)$. The tables are initially empty, once the adversary makes a query to an oracle that does not exist in the tables, the simulator inserts the query and the simulated answer to the table corresponding to the oracle.

Then, we describe the responses of simulator $\mathcal{S}$ to the queries at the adversarial interface.For any query, if the corresponding answer can be found in the tables maintained by $\mathcal{S}$, no matter the answer is exactly in the corresponding table or can be extrapolated from the associated items in different tables and the honest interface, the simulator will give an answer consistent with the tables by simply searching the tables and/or querying the honest interface with the corresponding inputs. And for any fresh query whose answer cannot be simply found using the tables, the simulator answer them as following:

For a fresh query SK_0 to the oracle H_0, if $\exists(*, *, \text{pk}_0, \text{PK}_0) \in T_{P_0^{-1}}$ s.t. $\text{PK}_0 = \mathsf{I.asyKG}_0(\text{SK}_0)$, sample random $\text{sk}_0 \leftarrow \{0,1\}^{n_1(\lambda)}$, add $(\text{SK}_0, \text{sk}_0, \text{pk}_0, \text{PK}_0)$ to T_{H_0}, and return sk_0. Otherwise, sample random $\text{sk}_0 \leftarrow \{0,1\}^{n_1(\lambda)}$, $\text{pk}_0 \leftarrow \{0,1\}^{n_2(\lambda)}$; query $\mathsf{I.asyKG}_0$ with SK_0 and obtain PK_0; add $(\text{SK}_0, \text{sk}_0, \text{pk}_0, \text{PK}_0)$ to T_{H_0}, and return sk_0. For a fresh query SK_1 to the oracle H_1, if $\exists(*, *, \text{pk}_1, \text{PK}_1) \in T_{P_1^{-1}}$ s.t. $\text{PK}_1 = \mathsf{I.asyKG}_1(\text{SK}_1)$, sample random $\text{sk}_1 \leftarrow \{0,1\}^{n_1(\lambda)}$, add $(\text{SK}_1, \text{sk}_1, \text{pk}_1, \text{PK}_1)$ to T_{H_1}, and return sk_1. Otherwise, $\text{sk}_1 \leftarrow \{0,1\}^{n_1(\lambda)}$, $\text{pk}_1 \leftarrow \{0,1\}^{n_2(\lambda)}$, query the external $\mathsf{I.asyKG}_1$ with SK_1 to obtain PK_1; add $(\text{SK}_1, \text{sk}_1, \text{pk}_1, \text{PK}_1)$ to T_{H_1}, and return sk_1.

For a fresh query pk_0 to the oracle P_0, randomly sample $\text{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$ and $\text{sk}_0 \leftarrow \{0,1\}^{n_1(\lambda)}$, query $\mathsf{I.asyKG}_0$ with SK_0 to obtain PK_0, return PK_0, and add $(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0)$ to the table T_{P_0}. For a fresh query PK_0 to the oracle P_0^{-1}, return a randomly sampled $\mathrm{pk}_0 \leftarrow \{0,1\}^{n_2(\lambda)}$, add $(*, *, \mathrm{pk}_0, \mathrm{PK}_0)$ to the table $T_{P_0^{-1}}$. For a fresh query pk_1 to the oracle P_1, randomly sample $\mathrm{SK}_1 \leftarrow \{0,1\}^{\ell_3(\lambda)}$ and $\mathrm{sk}_1 \leftarrow \{0,1\}^{n_1(\lambda)}$, query $\mathsf{I.asyKG_1}$ with SK_1 to obtain PK_1, return PK_1 and add $(\mathrm{SK}_1, \mathrm{sk}_1, \mathrm{pk}_1, \mathrm{PK}_1)$ to the table T_{P_1}. For a fresh query PK_1 to the oracle P_0^{-1}, return a randomly sampled $\mathrm{pk}_1 \leftarrow \{0,1\}^{n_2(\lambda)}$, add $(*, *, \mathrm{pk}_1, \mathrm{PK}_1)$ to the table $T_{P_1^{-1}}$.

For a fresh query sk to the oracle $\mathsf{I.symKG}$, randomly sample $\tilde{\mathrm{pk}} \leftarrow \{0,1\}^{n_2(\lambda)}$, $\tilde{\mathrm{SK}}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$ and $\tilde{\mathrm{SK}}_1 \leftarrow \{0,1\}^{\ell_3(\lambda)}$; query $\mathsf{I.asyKG_0}$ with $\tilde{\mathrm{SK}}_0$ to obtain $\tilde{\mathrm{PK}}_0$; query $\mathsf{I.asyKG_1}$ with $\tilde{\mathrm{SK}}_1$ to obtain $\tilde{\mathrm{PK}}_1$; return $\tilde{\mathrm{pk}}$, finally, add $(\tilde{\mathrm{SK}}_0, \mathrm{sk}, \tilde{\mathrm{pk}}, \tilde{\mathrm{PK}}_0)$ and $(\tilde{\mathrm{SK}}_1, \mathrm{sk}, \tilde{\mathrm{pk}}, \tilde{\mathrm{PK}}_1)$ to the table T_{symKG}.

For a fresh query $(\widehat{\mathrm{sk}}, \widetilde{\mathrm{pk}})$ to the oracle $\mathsf{I.symSHK}$, if there exists a PK_1 corresponding to $\widetilde{\mathrm{pk}}$ in the tables, randomly sample $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$, query $\mathsf{I.asySHK_0}$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$ to obtain K, and return K; if there exists a PK_0 corresponding to $\widetilde{\mathrm{pk}}$ in the tables, randomly sample $\mathrm{SK}_1 \leftarrow \{0,1\}^{\ell_3(\lambda)}$, query $\mathsf{I.asySHK_1}$ with $(\mathrm{SK}_1, \mathrm{PK}_0)$ to obtain K, and return K; if there exists a SK_0 corresponding to $\widehat{\mathrm{sk}}$ in the tables, randomly sample $\mathrm{PK}_1 \leftarrow \{0,1\}^{\ell_4(\lambda)}$, query $\mathsf{I.asySHK_0}$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$ to obtain K_0, and return K_0; if there exists a SK_1 corresponding to $\widehat{\mathrm{sk}}$ in the tables, randomly sample $\mathrm{PK}_0 \leftarrow \{0,1\}^{\ell_2(\lambda)}$, query $\mathsf{I.asySHK_1}$ with $(\mathrm{SK}_1, \mathrm{PK}_0)$ to obtain K_1, and return K_1; otherwise, randomly sample $\mathrm{SK}_1 \leftarrow \{0,1\}^{\ell_3(\lambda)}$ and $\tilde{\mathrm{PK}}_0 \leftarrow \{0,1\}^{\ell_2(\lambda)}$, query $\mathsf{I.asySHK_1}$ with $(\mathrm{SK}_1, \tilde{\mathrm{PK}}_0)$ to obtain K, return K, and finally, store all the newly generated query-and-answer items in the corresponding tables.

According to the output randomness of all the oracles on the adversarial interface and the relationship between the outputs and inputs of the oracles determined by the construction $\mathit{\Pi}.\mathsf{asyNIKE}$, the answers simulated by the simulator and that of the corresponding oracles in the real scheme are computationally indistinguishable for any PPT differentiator. Therefore, the constructed $\mathit{\Pi}.\mathsf{asyNIKE}$ is indistinguishable from an ideal asymmetric NIKE in the indifferentiability framework. The full proof of Theorem 1 is shown in Appendix A.

3.2 Indifferentiable Endemic OT from Ideal Asymmetric NIKE

We construct an indifferentiable EOT from an ideal asymmetric NIKE as below, where $H_0 : \mathcal{Z} \mapsto \mathcal{Z}$, $H_1 : \mathcal{Z} \mapsto \mathcal{Z}$, $H_2 : \mathcal{X} \times \{0,1\} \mapsto \mathcal{Y}$, $H_3 : \mathcal{Y} \mapsto \mathcal{Y}$ are random oracles, $\mathcal{E} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathcal{Y}$, $\mathcal{E}^{-1} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathcal{Y}$ is an ideal cipher.

- $\underline{Q \leftarrow \mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}(b, \mathrm{SK}_0):}$
  $\mathrm{PK}_0 \leftarrow \mathsf{I.asyKG_0}(\mathrm{SK}_0)$, $Q_{1-b} = H_2(\mathrm{SK}_0, b)$, $Q_b = \mathcal{E}(H_3(Q_{1-b}), \mathrm{PK}_0)$, output $Q = Q_0 \| Q_1$.
- $\underline{\mathrm{PK}_1 \leftarrow \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1) = \mathsf{I.asyKG_1}(\mathrm{SK}_1)}$
- $\underline{K_b \leftarrow \mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{PK}_1, \mathrm{SK}_0, b) = H_b(\mathsf{I.asySHK_0}(\mathrm{PK}_1, \mathrm{SK}_0))}$.

– $\underline{(K_0, K_1) \leftarrow \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}(Q, \mathrm{SK}_1)}$:
$Q = Q_0, Q_1,\ A_0 = \mathcal{E}^{-1}(H_3(Q_1), Q_0),\ A_1 = \mathcal{E}^{-1}(H_3(Q_0), Q_1)$,
$K_0 \leftarrow H_0(\mathsf{I.asySHK}_1(A_0, \mathrm{SK}_1)),\ K_1 \leftarrow H_1(\mathsf{I.asySHK}_1(A_1, \mathrm{SK}_1))$,
output (K_0, K_1).

We prove the security of the EOT construction below.

Theorem 2. *The constructed protocol $\mathit{\Pi}.\mathsf{EOT}$ in Sec. 3.2, with access to an ideal asymmetric NIKE scheme $\mathsf{I.asyNIKE}$, random oracles H_0, H_1, H_2, H_3 and an ideal cipher $\mathcal{E}$, is indifferentiable from an ideal EOT protocol $\mathsf{I.EOT}$ as in Def.5. More precisely, there exists a simulator $\mathcal{S}$ such that for all polynomial q-query distinguisher $\mathcal{D}$, the distinguishing advantage $\mathrm{Adv}^{\mathsf{indif}}_{\Pi.\mathsf{EOT},\mathsf{I.EOT},\mathcal{S},\mathcal{D}}(1^\lambda)$ satisfies the following:*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi.\mathsf{EOT},\mathsf{I.EOT},\mathcal{S},\mathcal{D}}(1^\lambda) \leq \frac{q^2}{2^{\ell_1(\lambda)}} + \frac{2q^2}{2^{\ell_2(\lambda)}} + \frac{q^2}{2^{n_3(\lambda)}} \leq \mathsf{negl}(\lambda),$$

Proof. In the real world, the differentiator $\mathcal{D}$ has oracle access to $(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2})$ via the honest interface and oracle access to $(H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$ via the adversarial interface. In contrast, in the ideal world, the differentiator $\mathcal{D}$ and the simulator $\mathcal{S}$has oracle access to $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface.

First, $\mathcal{S}$ maintains a table of queries and answers for each of the oracles $(H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$. The simulator maintains a table for each of the oracles, respectively, in the following form: $T_{H_0} = (\mathrm{SK}_0, b, Q_{1-b}, Q_b)$, $T_{H_1} = (Q_d, \tilde{k})$, $T_{\mathcal{E}} = (Q_b, \tilde{k}, \mathrm{PK}_0)$, $T_{\mathcal{E}^{-1}} = (Q_b, \tilde{k}, \mathrm{PK}_0)$, $T_{\mathrm{asyKG}_0} = (\mathrm{SK}_0, \mathrm{PK}_0)$, $T_{\mathrm{asyKG}_1} = (\mathrm{SK}_1, \mathrm{PK}_1)$, $T_{\mathrm{asySHK}_0} = (\mathrm{SK}_0, \mathrm{PK}_1, K)$, $T_{\mathrm{asySHK}_1} = (\mathrm{SK}_1, \mathrm{PK}_0, K)$.

Then the simulator $\mathcal{S}$ responds to oracle queries at the adversarial interface using the similar strategies as that in the proof of Theorem 1. In particular, for any query, if the corresponding answer can be found by searching the tables and/or querying the honest interface, $\mathcal{S}$ answers it using these methods. And for any fresh query whose answer cannot be simply found with those methods, the simulator answer them as following:

For a fresh query (SK_0, b) at H_0, the simulator will query the external $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK_0, b) to obtain Q_0, Q_1, reply with Q_{1-b}, and store $(\mathrm{SK}_0, b, Q_{1-b}, Q_b)$ in the table T_{H_0}. For a fresh query Q_d at H_1, the simulator will sample a random $\tilde{k}$ from the range of H_1, reply with $\tilde{k}$ and store $(Q_d, \tilde{k})$ in the table T_{H_1}. For a fresh query $(\mathrm{PK}_0, \tilde{k})$ at $\mathcal{E}$, the simulator will uniformly sample a SK_0 from $\{0,1\}^{\ell_1(\lambda)}$, and a bit $b \leftarrow \{0,1\}$, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK_0, b) to obtain (Q_0, Q_1), reply with Q_b, and store $(Q_{1-b}, \tilde{k})$, $(\mathrm{SK}_0, \mathrm{K}_0)$ and $(Q_b, \tilde{k}, \mathrm{PK}_0)$ in the table T_{H_1}, T_{asyKG_0} and $T_{\mathcal{E}}$, respectively. For a fresh query $(Q_b, \tilde{k})$ at $\mathcal{E}^{-1}$, the simulator will uniformly sample a PK_0 from $\{0,1\}^{\ell_2(\lambda)}$,, reply with PK_0, store $(Q_b, \tilde{k}, \mathrm{PK}_0)$ in the table $T_{\mathcal{E}^{-1}}$. For a fresh query SK_0 at $\mathsf{I.asyKG}_0$, the simulator will uniformly sample a PK_0 from $\{0,1\}^{\ell_2(\lambda)}$,, reply with PK_0, store $(\mathrm{SK}_0, \mathrm{PK}_0)$ in the table T_{asyKG_0}. For a fresh query SK_1 at $\mathsf{I.asyKG}_1$, the simulator will query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}$

with SK_1, obtain PK_1, reply with PK_1, store $(\mathrm{SK}_1, \mathrm{PK}_1)$ in the table T_{asyKG_1}. The simulated answers are indistinguishable from the answers of the real oracles at the adversarial interfaces. Therefore, the real world construction $\Pi.\mathsf{EOT}$ is indifferentiable from an ideal one-round EOT.

The full proof of Theorem 2 is shown in Appendix B.

3.3 NIKE from EOT in Indifferentiability Framework

Using $\mathsf{I.EOT}$ as the main component, we construct a NIKE scheme $\Pi.\mathsf{symNIKE} = (\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$, described as following, where H_1 is a random oracle, H_0 is a random function defined below: $H_1 : \{0,1\}^* \mapsto \{0,1\}^{\ell_3(\lambda)}$, $H_0 : \{0,1\}^{n_3(\lambda)} \times \{0,1\}^{n_3(\lambda)} \mapsto \{0,1\}^{n_3(\lambda)}$.

- $\underline{\Pi.\mathsf{symKG}(\mathrm{SK}_0) \to \mathrm{PK}_0}$:
  $\mathrm{PK}_0^A \leftarrow \mathsf{I.EOT}_{A_1}(\mathrm{SK}_0, 0)$, $\mathrm{PK}_0^B \leftarrow \mathsf{I.EOT}_{B_1}(H_1(\mathrm{SK}_0))$,
  output $\mathrm{PK}_0 = (\mathrm{PK}_0^A, \mathrm{PK}_0^B)$.

- $\underline{\Pi.\mathsf{symSHK}(\mathrm{SK}_0, \mathrm{PK}_1) \to K}$:
  $K = H_0(\mathsf{I.EOT}_{A_2}(\mathrm{SK}_0, \mathrm{PK}_1^B, 0), \mathsf{LoR}_0(\mathsf{I.EOT}_{B_2}(H_1(\mathrm{SK}_0), \mathrm{PK}_1^A))$,[7]
  output K.

By the definition of the ideal NIKE, the correctness holds. We argue the security of $\Pi.\mathsf{symNIKE}$ below.

Theorem 3. *The constructed scheme* $\Pi.\mathsf{symNIKE}$ *in Sec. 3.3, with access to an ideal EOT protocol* $\mathsf{I.EOT}$ *and random oracles* H_0, H_1*, is indifferentiable from an ideal symmetric NIKE scheme as in Def. 2. More precisely, there exists a simulator* $\mathcal{S}$ *such that for all polynomial q-query distinguisher* $\mathcal{D}$*, we have*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi.\mathsf{symNIKE},\mathsf{I.symNIKE},\mathcal{S},\mathcal{D}}(1^\lambda) \leq \frac{q^2}{2^{\ell_2(\lambda)}} + \frac{2q^2}{2^{\ell_3(\lambda)}} + \frac{5q^2}{2^{n_3(\lambda)}} + \frac{q^2}{2^{2n_3(\lambda)}} \leq \mathsf{negl}(\lambda),$$

Proof. In the real world, the differentiator $\mathcal{D}$ has oracle access to $(\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$ via the honest interface and oracle access to $(H_0, H_1, \mathsf{I.EOT}_{A_1}, \mathsf{I.EOT}_{A_2}, \mathsf{I.EOT}_{B_1}, \mathsf{I.EOT}_{B_2})$ via the adversarial interface. In contrast, in the ideal world, the differentiator $\mathcal{D}$ has oracle access to $(\mathsf{I.symKG}, \mathsf{I.symSHK})$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface. Therefore, to establish a proof, we build an explicit (and efficient) simulator $\mathcal{S}$ that simulates the rest oracles $(H_0, H_1, \mathsf{I.EOT}_{A_1}, \mathsf{I.EOT}_{A_2}, \mathsf{I.EOT}_{B_1}, \mathsf{I.EOT}_{B_2})$ properly by making queries to $(\mathsf{I.symKG}, \mathsf{I.symSHK})$.

The simulator maintains a table for each of those oracle in the following forms, where the subscript indicates the corresponding oracle: $T_{\mathsf{H}_0} = (\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0)$, $T_{\mathsf{H}_1} = (\mathrm{SK}_1, \mathrm{sk}_1, \mathrm{pk}_1, \mathrm{PK}_1)$, $T_{\mathsf{P}_0} = (\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0)$, $T_{\mathsf{P}_0^{-1}} = (*, *, \mathrm{pk}_0, \mathrm{PK}_0)$, $T_{\mathsf{P}_1} = (\mathrm{SK}_1, \mathrm{sk}_1, \mathrm{pk}_1, \mathrm{PK}_1)$, $T_{\mathsf{P}_1^{-1}} = (*, *, \mathrm{pk}_1, \mathrm{PK}_1)$,

[7] the two inputs of H_0 are in lexicographical order

$T_{\mathsf{symKG}} = (*, \mathrm{sk}, \mathrm{pk}, *)$, $T_{\mathsf{symSHK}} = (\widehat{\mathrm{sk}}, \tilde{\mathrm{pk}}, K)$. The tables are initially empty, once the adversary makes a query to an oracle that does not exist in the tables, the simulator inserts the query and the simulated answer to the table corresponding to the oracle. The simulator answers the queries at the adversarial interface as follows. For any query, if the corresponding answer can be found by searching the tables maintained by $\mathcal{S}$ and/or querying the honest interface, $\mathcal{S}$ answers it using these methods. And for any fresh query whose answer cannot be simply found using those methods, the simulator answer them as follows.

For a fresh query SK to the oracle H_1, return a randomly sampled $\overline{\mathrm{SK}} \leftarrow \{0,1\}^{\ell_1(\lambda)}$; add $(\mathrm{SK}, \overline{\mathrm{SK}})$ to T_{H_1}. For a fresh query $(\mathrm{SK}, 1)$ to the oracle $\mathsf{I.EOT}_{\mathsf{A}_1}$, return a randomly sampled $\mathrm{PK}^L \leftarrow \{0,1\}^{2\ell_5(\lambda)}$, and add $(\mathrm{SK}, 1, \mathrm{PK}^L)$ to the table $T_{\mathrm{OT}_{A_1}}$. For a fresh query $(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R)$ to the oracle $\mathsf{I.EOT}_{\mathsf{A}_2}$, return a randomly sampled $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$, and add $(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R, K_A)$ to $T_{\mathrm{OT}_{A_2}}$. For a fresh query $\overline{\mathrm{SK}}$ to $\mathsf{I.EOT}_{\mathsf{B}_1}$, randomly sample $\mathrm{SK} \leftarrow \{0,1\}^{\ell_1(\lambda)}$, query $\mathsf{I.symKG}$ with SK to obtain PK (which has $\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits of PK as PK^R; return PK^R; add $(\mathrm{SK}, \overline{\mathrm{SK}})$ to T_{H_1} and $(\overline{\mathrm{SK}}, \mathrm{PK}^R)$ to $T_{\mathrm{OT}_{B_1}}$. For a fresh query $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)$ to $\mathsf{I.EOT}_{\mathsf{B}_2}$, if there exists the corresponding answer's first half K_B or the second half K_A in the tables, randomly sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$, return $K_B, \tilde{K_B}$ or $\tilde{K_B}, K_A$ and add $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B})$ or $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, \tilde{K_B}, K_A)$ to the table $T_{\mathrm{OT}_{B_2}}$. Otherwise, return randomly sampled $K_B, \tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$, and add $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B})$ to the table $T_{\mathrm{OT}_{B_2}}$. For a fresh query (K_A, K_B) to the oracle H_0, return a random sampled $K \leftarrow \{0,1\}^{n_3(\lambda)}$; add (K_A, K_B, K) to the table T_{H_0}.

The simulated answers are indistinguishable from the answers of the real oracles at the adversarial interfaces. Therefore, the real world construction $\Pi.\mathsf{symNIKE}$ is indifferentiable from an ideal (symmetric) NIKE. The full proof of Theorem 3 is provided in Appendix C.

4 Relations between two-round OT and PKE in Indifferentiability Framework

4.1 Indifferentiable PKE from Ideal Two-Round OT

First, we present the definition of ideal two-round OT as follows.

Definition 6. ***[Ideal Two-Round OT]*** *Let $\mathcal{X}_1, \mathcal{Y}, \mathcal{M}, \mathcal{X}_2, \mathcal{Z}, \mathcal{W} \in \omega(\log \lambda)$ be five sets. Let $\mathcal{F}_1 := \{\{0,1\} \times \mathcal{X}_1 \mapsto \mathcal{Y}\}$ be a family of functions that maps an element in $\{0,1\} \times \mathcal{X}_1$ to an element in $\mathcal{Y}$. Let $\mathcal{F}_2 := \{\mathcal{Y} \times \mathcal{M} \times \mathcal{M} \times \mathcal{X}_2 \mapsto \mathcal{W}\}$ be a family of functions that maps an element in $\mathcal{Y} \times \mathcal{M} \times \mathcal{M} \times \mathcal{X}_2$ to an element in $\mathcal{W}$. Let $\mathcal{F}_3 := \{\mathcal{W} \times \mathcal{X}_1 \times \{0,1\} \mapsto \mathcal{M}\}$ be a family of functions that maps an element in $\mathcal{W} \times \mathcal{X}_1 \times \{0,1\}$ to an element in $\mathcal{M}$.*

We define ideal two-round OT $\mathsf{I.2OT} = (\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3)$ as the set of all function tuples (f_1, f_2, f_3) such that: (1) for $i \in [3]$, $f_i \in \mathcal{F}_i$;

and (2) $\forall b \in \{0,1\}, \forall x_1 \in \mathcal{X}_1, \forall x_2 \in \mathcal{X}_2, \forall m_0, m_1 \in \mathcal{M}$*, it holds that* $f_3(f_2(f_1(b, x_1)), m_0, m_1, x_2) = m_b$.

In this section, we ignore the parameters of these ideal objects for simplicity, without influencing our theorems.

We construct a PKE scheme without public key validity test, denoted by $\Pi.\mathsf{PKE} = (\Pi.\mathsf{KGEN}, \Pi.\mathsf{ENC}, \Pi.\mathsf{DEC})$, from an ideal two-round OT protocol $\mathsf{I.2OT} = (\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3)$, as following.

- $\mathrm{PK} \leftarrow \Pi.\mathsf{KGEN}(\mathrm{SK}) = \mathsf{I.OT}_1(0, \mathrm{SK})$
- $C = \Pi.\mathsf{ENC}(\mathrm{PK}, m, \overline{\mathrm{SK}}) = \mathsf{I.OT}_2(m, m, \mathrm{PK}, \overline{\mathrm{SK}})$, where $\overline{\mathrm{SK}}$ serves as the nonce for encryption.
- $m = \Pi.\mathsf{DEC}(\mathrm{SK}, C) = \mathsf{I.OT}_3(C, \mathrm{SK}, 0)$

By definition, the correctness holds. We argue the security of $\Pi.\mathsf{PKE}$ below.

Theorem 4. *The constructed scheme* $\Pi.\mathsf{PKE}$ *in Sec. 4.1, with access to an ideal two-round OT protocol* $\mathsf{I.2OT}$*, is indifferentiable from an ideal PKE scheme as in Def. 3.*

Proof. In the real world, the differentiator $\mathcal{D}$ has oracle access to ($\Pi.\mathsf{KGEN}$, $\Pi.\mathsf{ENC}$, $\Pi.\mathsf{DEC}$) via the honest interface and oracle access to ($\mathsf{I.OT}_1$, $\mathsf{I.OT}_2$, $\mathsf{I.OT}_3$) via the adversarial interface. In the ideal world, the differentiator $\mathcal{D}$ has oracle access to ($\mathsf{I.KGEN}$, $\mathsf{I.ENC}$, $\mathsf{I.DEC}$) via the honest interface and access to $\mathcal{S}$ via the adversarial interface.

Note that the inputs for the adversarial interface and for the honest interface are highly matched in both the ideal world and the real world. For most of the queries, the simulator $\mathcal{S}$ can simply use the full or partial query input(s) to query the honest interface and obtain an output, then use the output as the response. For example, for a query $(0, \mathrm{SK})$ to the oracle $\mathsf{I.OT}_1$, the simulator just queries $\mathsf{I.KGEN}$ with SK to obtain PK, and return PK as the answer. In this way, $\mathcal{S}$'s responses are consistent with the honest interface. For queries not covered in the above, the simulator answers them as follows. For a fresh query $(\mathrm{SK}, 1)$ to the oracle $\mathsf{I.OT}_1$, return a randomly sampled $\widetilde{\mathrm{PK}} \leftarrow \mathsf{Y}$, and add $(\mathrm{SK}, b, \widetilde{\mathrm{PK}})$ to the table T_{OT_1}. For a fresh query $(m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}})$ (with $m_0 \neq m_1$) to $\mathsf{I.OT}_2$, return a randomly sampled $C \leftarrow \mathcal{C}$ and add $(m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}}, C)$ to the table T_{OT_2}. For a fresh query $(\mathrm{SK}, 1, C)$ to $\mathsf{I.OT}_3$, return a randomly sampled $\tilde{m} \leftarrow \mathcal{M}$ and add $(\mathrm{SK}, 1, C, \tilde{m})$ to the table T_{OT_3}.

Since the outputs of the oracles $\mathsf{I.OT}_1$, $\mathsf{I.OT}_2$, $\mathsf{I.OT}_3$ are randomly distributed, the simulator's responses are indistinguishable from their responses. Therefore, the constructed PKE scheme $\Pi.\mathsf{PKE}$ is indifferentiable from an ideal PKE. The full proof can be found in Appendix D.1.

4.2 Indifferentiable two-round OT from Ideal PKE Without Public Key Validity Test

We construct a two-round OT protocol, denoted by $\Pi.\mathsf{2OT} = (\Pi.\mathsf{OT}_1, \Pi.\mathsf{OT}_2, \Pi.\mathsf{OT}_3)$, from an ideal PKE $\mathsf{PKE} = (\mathsf{KGEN}, \mathsf{ENC}, \mathsf{DEC}))$, as following, where H_0, H_1, H_2, H_3 are random oracles, (P_0, P_0^{-1}) are an ideal permutation and its inverse, and $(\mathcal{E}, \mathcal{E}^{-1})$ are an ideal cipher and its inverse.

– $\underline{Q \leftarrow \mathit{\Pi}.\mathsf{OT}_1(b, \mathrm{SK})}$:
  $\mathrm{PK} = \mathsf{KGEN}(\mathrm{SK})$, $Q_{1-b} = H_2(\mathrm{SK}, b)$, $Q_b = \mathcal{E}(H_3(Q_{1-b}), \mathrm{PK})$, $Q = Q_0 \| Q_1$

– $\underline{w \leftarrow \mathit{\Pi}.\mathsf{OT}_2(Q, m_0, m_1, \overline{\mathrm{SK}})}$
  $\mathrm{PK}_0 := \mathcal{E}^{-1}(H_3(Q_1), Q_0)$, $\mathrm{PK}_1 := \mathcal{E}^{-1}(H_3(Q_0), Q_1)$,
  $C_0 \leftarrow \mathsf{ENC}(\mathrm{PK}_0, m_0, H_0(\overline{\mathrm{SK}}, m_1))$, $C_1 \leftarrow \mathsf{ENC}(\mathrm{PK}_1, m_1, H_1(\overline{\mathrm{SK}}, m_0))$,
  $w := P_0(C_0, C_1)$

– $\underline{m_b \leftarrow \mathit{\Pi}.\mathsf{OT}_3(w, \mathrm{SK}, b)}$
  $W = P_0^{-1}(w)$, $W = C_0, C_1$, $m_b = \mathsf{I.DEC}(\mathrm{SK}, C_b)$

By definition, the correctness holds. We show the security of this 2OT construction via Theorem 5, below.

Theorem 5. *The constructed protocol $\mathit{\Pi}$.2OT in Sec. 4.2, with access to an ideal PKE* I.PKE*, random oracles H_0, H_1, H_2, H_3, a random permutation P_0 and an ideal cipher $(\mathcal{E}, \mathcal{E}^{-1})$, is indifferentiable from an ideal two-round OT protocol as in Def. 6.*

Proof. In the real world, the differentiator $\mathcal{D}$ has oracle access to ($\mathit{\Pi}.\mathsf{OT}_1$, $\mathit{\Pi}.\mathsf{OT}_2$, $\mathit{\Pi}.\mathsf{OT}_3$ via the honest interface and oracle access to (H_0, H_1, H_2, H_3, P_0, P_0^{-1}, $\mathcal{E}, \mathcal{E}^{-1}$ via the adversarial interface. In the ideal world, the differentiator $\mathcal{D}$ has oracle access to ($\mathsf{I.OT}_1$, $\mathsf{I.OT}_2$, $\mathsf{I.OT}_3$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface.

The simulator $\mathcal{S}$ maintains a table for each oracle at the adversarial interface, in the following forms: $T_{H_0} = (\overline{\mathrm{SK}}, m_1, r_0)$, $T_{H_1} = (\overline{\mathrm{SK}}, m_0, r_1)$, $T_{H_2} = (\mathrm{SK}, b, Q_{1-b})$, $T_{H_3} = (Q, K)$, $T_{P_0} = (C_0, C_1, w)$, $T_{P_0^{-1}} = (C_0, C_1, w)$, $T_{\mathcal{E}} = (K, \mathrm{PK}, Q)$, $T_{\mathcal{E}^{-1}} = (K, \mathrm{PK}, Q)$.

The simulator $\mathcal{S}$ in the ideal world is described below. For any query, if the corresponding answer can be found by searching the tables and/or querying the honest interface, $\mathcal{S}$ answers it using these methods. And for any fresh query whose answer cannot be simply found with those methods, the simulator answer them as following: For a fresh query $(\overline{\mathrm{SK}}, m_1)$ to the oracle H_0, return a randomly sampled $r_0 \leftarrow \mathcal{R}$; add $(\overline{\mathrm{SK}}, m_1, r_0)$ to the table T_{H_0}. For a fresh query $(\overline{\mathrm{SK}, m_0})$ to the oracle H_1, return a randomly sampled $r_1 \leftarrow \mathcal{R}$; add $(\overline{\mathrm{SK}}, m_0, r_1)$ to the table T_{H_1}. For a fresh query Q to the oracle H_3, return a randomly sampled $K \leftarrow \mathcal{K}$; add (Q, K) to the table T_{H_3}. For a fresh query (K, PK) to the oracle $\mathcal{E}$, return a randomly sampled $Q \leftarrow \mathcal{C}$; add (K, PK, Q) to the table $T_{\mathcal{E}}$. For a fresh query (K, Q) to the oracle $\mathcal{E}^{-1}$, return a randomly sampled $\mathrm{PK} \leftarrow \mathcal{PK}$; add (K, PK, Q) to the table $T_{\mathcal{E}^{-1}}$. For a fresh query $(C_0 \| C_1)$ to the oracle P_0, return a randomly sampled $w \leftarrow \mathcal{C} \times \mathcal{C}$; add (C_0, C_1, w) to T_{P_0}. For a fresh query w to the oracle P^{-1}, return randomly sampled $C_0, C_1 \leftarrow \mathcal{C}$; add (C_0, C_1, w) to $T_{P_0^{-1}}$. For a fresh query SK to the oracle I.KGEN, return a randomly sampled $\mathrm{PK} \leftarrow \mathcal{PK}$;

add $(\mathrm{SK}, \mathrm{PK})$ to T_{KGen}. For a fresh query (PK, m, r) to the oracle $\mathsf{I.ENC}$, return a randomly sampled $C \leftarrow \mathcal{C}$; add (PK, m, r, C) to the table T_{Enc}. For a fresh query (SK, C) to the oracle $\mathsf{I.DEC}$, return a randomly sampled $m \leftarrow \mathcal{M}$; add (SK, C, m) to the table T_{Dec}.

The simulated answers are indistinguishable from the answers of the real oracles at the adversarial interfaces. Therefore, the real world construction $\Pi.\mathsf{2OT}$ is indifferentiable from an ideal two round OT. The full proof can be found in Appendix D.2.

5 The relationship between Ideal one-round EOT and Ideal two-round OT

Here we provide a construction of two-round OT $\Pi.\mathsf{2OT} = (\Pi.\mathsf{OT}_1, \Pi.\mathsf{OT}_2, \Pi.\mathsf{OT}_3)$ based on an ideal one-round EOT $\mathsf{I.EOT}$, where H_0 is a random oracle, P_0 is a random permutation, $(\mathcal{E}_1, \mathcal{E}_1^{-1})$ and $(\mathcal{E}_2, \mathcal{E}_2^{-1})$ are ideal ciphers.

- $\underline{Q \leftarrow \Pi.\mathsf{OT}_1(\mathrm{SK}, b) = \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, b)}$

- $\underline{w \leftarrow \Pi.\mathsf{OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})}$
  $e = H_0(Q, m_0, m_1, \widetilde{\mathrm{SK}})$, $\widetilde{\mathrm{PK}} \leftarrow \mathsf{I.EOT}_{\mathsf{B}_1}(\widetilde{\mathrm{SK}})$, $\widehat{\mathrm{PK}} = \mathcal{E}_1(e, \widetilde{\mathrm{PK}})$,
  $(K_0, K_1) \leftarrow \mathsf{I.EOT}_{\mathsf{B}_2}(Q, \widetilde{\mathrm{SK}})$, $C_0 = \mathcal{E}_2(K_0, m_0)$, $C_1 = \mathcal{E}_2(K_1, m_1)$,
  $w = P_0(\widehat{\mathrm{PK}}, e, C_0, C_1)$

- $\underline{m_b \leftarrow \Pi.\mathsf{OT}_3(w, b, \mathrm{SK})}$
  $(\widehat{\mathrm{PK}}, e, C_0, C_1) = P_0^{-1}(w)$, $\widetilde{\mathrm{PK}} = \mathcal{E}_1^{-1}(e, \widehat{\mathrm{PK}})$, $K_b \leftarrow \mathsf{I.EOT}_{\mathsf{A2}}(\widetilde{\mathrm{PK}}, b, \mathrm{SK})$,
  $m_b = \mathcal{E}_2^{-1}(K_b, C_b)$.

Theorem 6. *The constructed protocol $\Pi.\mathsf{2OT}$ in Sec. 5, with access to an ideal one-round EOT protocol $\mathsf{I.EOT}$, a random oracle H_0, a random permutation P_0 and two ideal ciphers $(\mathcal{E}_1, \mathcal{E}_1^{-1})$, $(\mathcal{E}_2, \mathcal{E}_2^{-1})$, is indifferentiable from an ideal two-round OT protocol as in Def. 6.*

Proof. In the real world, the differentiator $\mathcal{D}$ has oracle access to $(\Pi.\mathsf{OT}_1, \Pi.\mathsf{OT}_2, \Pi.\mathsf{OT}_3)$ via the honest interface and oracle access to $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{EOT}_{\mathsf{B}_2})$ via the adversarial interface. In the ideal world, the differentiator $\mathcal{D}$ has oracle access to $(\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface.

First, the simulator $\mathcal{S}$ maintains a table for each oracle at the adversarial interface as in the previous proofs. Then for any query, if the corresponding answer can be found by searching the tables and/or querying the honest interface, $\mathcal{S}$ answers it using these methods. And for any fresh query whose answer cannot be simply found with those methods, the simulator answer them as following:

For a fresh query $(Q, m_0, m_1, \widetilde{\mathrm{SK}})$ to the oracle H_0, query the external $\mathsf{I.OT}_2$ with $(Q, m_0, m_1, \widetilde{\mathrm{SK}})$ to obtain e; return e, and add $(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e)$ to the table T_{H_0}. For a fresh query $(e, \widetilde{\mathrm{PK}})$ to the oracle $\mathcal{E}_1$, return a randomly sampled $\widehat{\mathrm{PK}}$; add $(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}})$ to the table $T_{\mathcal{E}_1}$. For a fresh query $(e, \widehat{\mathrm{PK}})$ to $\mathcal{E}_1^{-1}$, return a randomly sampled $\widetilde{\mathrm{PK}}$; add $(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}})$ to the table $T_{\mathcal{E}_1^{-1}}$. For a fresh query (K, m) to $\mathcal{E}_2$, return a randomly sampled C; add (K, m, C) to the table $T_{\mathcal{E}_2}$. For a fresh query (K, C) to $\mathcal{E}_2^{-1}$, return a randomly sampled m; add (K, m, C) to the table $T_{\mathcal{E}_2^{-1}}$. For a fresh query (SK, b) to $\mathsf{I.EOT}_{\mathsf{A}_1}$, query $\mathsf{I.OT}_1(\mathrm{SK}, b)$ to obtain Q; return Q, and add (SK, b, Q) to the table $T_{\mathrm{EOT}_{\mathrm{A}_1}}$. For a fresh query $(\widetilde{\mathrm{SK}})$ to $\mathsf{I.EOT}_{\mathsf{B}_1}$, return a randomly sampled $\widetilde{\mathrm{PK}}$; add $(\widetilde{\mathrm{SK}}, \widetilde{\mathrm{PK}})$ to the table $T_{\mathrm{EOT}_{\mathrm{B}_1}}$. For a fresh query $(\mathrm{SK}, b, \widetilde{\mathrm{PK}})$ to $\mathsf{I.EOT}_{\mathsf{A}_2}$, return a randomly sampled K_b; add $(\mathrm{SK}, b, \widetilde{\mathrm{PK}}, K_b)$ to the table $T_{\mathrm{EOT}_{\mathrm{A}_2}}$. For a fresh query $(Q, \widetilde{\mathrm{SK}})$ to $\mathsf{I.EOT}_{\mathsf{B}_2}$, return a randomly sampled K_0, K_1, add $(Q, \widetilde{\mathrm{SK}}, K_0, K_1)$ to the table $T_{\mathrm{EOT}_{\mathrm{B}_2}}$. For a fresh query $(\widehat{\mathrm{PK}}, e, C_0, C_1)$ to P_0, return a randomly sampled w, add $(\widehat{\mathrm{PK}}, e, C_0, C_1, w)$ to the table T_{P_0}. For a fresh query w to P_0^{-1}, return a randomly sampled $(\widehat{\mathrm{PK}}, e, C_0, C_1)$, add $(\widehat{\mathrm{PK}}, e, C_0, C_1, w)$ to the table $T_{P_0^{-1}}$.

The simulated answers are indistinguishable from the answers of the real oracles at the adversarial interfaces. Therefore, the real world construction $\mathit{\Pi}.\mathsf{2OT}$ is indifferentiable from an ideal two round OT. The full proof can be found in Appendix E.

Theorem 7. *Let* $\mathsf{I.2OT}$ *denote an ideal two-round OT protocol. For any construction of a one-round EOT protocol* $\mathit{\Pi}.\mathsf{EOT}$*, with access to the* $\mathsf{I.2OT}$ *and random oracles, there exists a PPT differentiator that can distinguish the constructed* $\mathit{\Pi}.\mathsf{EOT}$ *from an ideal one-round EOT protocol* $\mathsf{I.EOT}$ *as in Def. 5.*

Proof. Our constructed one-round EOT $\mathit{\Pi}.\mathsf{EOT}$ has four interfaces $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{A2}}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$, and an ideal two-round OT has three interfaces $\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3$. According to the interface parameters, $\mathsf{I.OT}_1$ should be used to construct $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}$, $\mathsf{I.OT}_2$ should be used to construct $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}$ and $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$, and $\mathsf{I.OT}_3$ should be used to construct $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A2}}$. Note that $\mathsf{I.OT}_2$ takes the output of $\mathsf{I.OT}_1$ as input to ensure correctness, however, $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}$ does not need the output of $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}$ as its input. Then the algorithm $\mathsf{I.OT}_2$ underlying $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$ cannot obtain the output of $\mathsf{I.OT}_1$ as its input, and the underlying $\mathsf{I.OT}_2$, $\mathsf{I.OT}_3$ cannot be computed correctly.

Therefore, the correctness of any constructed EOT based on $\mathsf{I.2OT}$ cannot be ensured, and there exists a PPT differentiator that can distinguish the constructed one-round EOT $\mathit{\Pi}.\mathsf{EOT}$ from an ideal EOT protocol, which completes our proof.

Discussion on the relationship between ideal PKE and ideal NIKE. Zhandry and Zhang [24] showed a construction of indifferentiable PKE without public key validity test from ideal NIKE without public key validity test. However, it is not clear whether ideal PKE implies ideal NIKE, after trying a lot of methods, we did not find a correct and secure construction of indifferentiable NIKE from ideal PKE. Based the our previous theorems, we obtain the following corollary.

Corollary 1. *According to Theorem 1, Theorem 2, Theorem 3, Theorem 4, Theorem 5, Theorem 6 and Theorem 7, ideal PKE does not imply ideal NIKE in the indifferentiablity framework.*

Proof. In the indifferentiablity framework, based on Theorem 1, Theorem 2 and Theorem 3, ideal NIKE is equivalent to ideal one-round EOT. And based on Theorem 4 and Theorem 5, ideal PKE is equivalent to ideal two-round OT. However, based on Theorem 6 and Theorem 7, ideal two round OT does not imply ideal one-round EOT; therefore, ideal PKE does not imply ideal NIKE in the indifferentiablity framework.

References

1. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indifferentiability of key-alternating ciphers. In: CRYPTO 2013. LNCS, vol. 8042, pp. 531–550. Springer (2013)
2. Barbosa, M., Farshim, P.: Indifferentiable authenticated encryption. In: CRYPTO 2018. LNCS, vol. 10991, pp. 187–220. Springer (2018)
3. Bellare, M., Micali, S.: Non-interactive oblivious transfer and spplications. In: CRYPTO, 1989. LNCS, vol. 435, pp. 547–557. Springer (1989)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS 1993. pp. 62–73. ACM (1993)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. pp. 136–145. IEEE Computer Society (2001)
6. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer (2005)
7. Coron, J., Dodis, Y., Mandal, A., Seurin, Y.: A domain extender for the ideal cipher. In: TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer (2010)
8. Coron, J., Holenstein, T., Künzler, R., Patarin, J., Seurin, Y., Tessaro, S.: How to build an ideal cipher: The indifferentiability of the feistel construction. J. Cryptol. **29**(1), 61–114 (2016)
9. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer (2009)
10. Dodis, Y., Stam, M., Steinberger, J.P., Liu, T.: Indifferentiability of confusion-diffusion networks. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 679–704. Springer (2016)
11. Friolo, D., Masny, D., Venturi, D.: A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In: TCC 2019. LNCS, vol. 11891, pp. 111–130. Springer (2019)
12. Garg, S., Ishai, Y., Srinivasan, A.: Two-round MPC: information-theoretic and black-box. In: TCC 2018. LNCS, vol. 11239, pp. 123–151. Springer (2018)
13. Garg, S., Srinivasan, A.: Garbled protocols and two-round MPC from bilinear maps. In: FOCS 2017. pp. 588–599. IEEE Computer Society (2017)
14. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: FOCS 2000. pp. 325–335. IEEE Computer Society (2000)

15. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC 1987. pp. 218–229. ACM (1987)
16. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology - EUROCRYPT 2008, Proceedings. LNCS, vol. 4965, pp. 415–432. Springer (2008)
17. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer (2008)
18. Li, S., Zhang, C., Lin, D.: Oblivious transfer from rerandomizable PKE. In: ICICS 2023. LNCS, vol. 14252, pp. 128–145. Springer (2023)
19. Masny, D., Rindal, P.: Endemic oblivious transfer. In: CCS 2019. pp. 309–326. ACM (2019)
20. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Proceedings. LNCS, vol. 2951, pp. 21–39. Springer (2004)
21. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer (2008)
22. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(4), 656–715 (1949)
23. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS 1982. pp. 160–164. IEEE Computer Society (1982)
24. Zhandry, M., Zhang, C.: Indifferentiability for public key cryptosystems. In: CRYPTO 2020. LNCS, vol. 12170, pp. 63–93. Springer (2020)
25. Zhou, Z., Zhang, B., Zhou, H., Ren, K.: Endemic oblivious transfer via random oracles, revisited. In: EUROCRYPT 2023,Part I. LNCS, vol. 14004, pp. 303–329. Springer (2023)

A Proof of Theorem 1

Proof. We use hybrid arguments to prove the indistinguishability of the constructed real asymmetric NIKE scheme from an ideal asymmetric NIKE scheme. First, we describe our simulator $\mathcal{S}$ in the hybrid games.

Game 0. This game is identical to the real game, namely, the simulator's responses are the same as in the real game, except that simulator records the eight type of queries and responses in the corresponding eight tables, referring to H_0 table T_{H_0}, H_1 table T_{H_1}, P_0 table T_{P_0}, P_1 table T_{P_1}, P_0^{-1} table $T_{\mathsf{P}_0^{-1}}$, P_1^{-1} table $T_{\mathsf{P}_1^{-1}}$, symKG table T_{symKG} and symSHK table T_{symSHK}, respectively.

In Game 0, the simulator maintains the tables as following.

1. H_0-table T_{H_0}: initially empty, consists of tuples with form of $(\mathrm{SK}_0, \mathrm{sk}_0, *, *)$. Once the adversary queries oracle H_0 with SK_0 which does not exist in T_{H_0}, the simulator inserts $(\mathrm{SK}_0, H_0(\mathrm{SK}_0), *, \Pi.\mathsf{asyKG}_0(\mathrm{SK}_0))$ into T_{H_0} table.
2. H_1-table T_{H_1}: initially empty, consists of tuples with form of $(\mathrm{SK}_1, \mathrm{sk}_1, *, *)$. Once the adversary queries oracle H_1 with SK_1 which does not exist in T_{H_1}, the simulator inserts $(\mathrm{SK}_1, H_1(\mathrm{SK}_1), *, \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1))$ into T_{H_1}-table.

3. P_0-table T_{P_0}: initially empty, consists of tuples with form of $(*, *, \mathrm{pk}_0, \mathrm{PK}_0)$. Once the adversary queries oracle P_0 with pk_0 which does not exist in T_{P_0}, the simulator inserts $(*, *, \mathrm{pk}_0, P_0(\mathrm{pk}_0))$ into P_0-table.
4. P_0^{-1}-table $T_{\mathsf{P}_0^{-1}}$: initially empty, consists of tuples with form of $(*, *, \mathrm{pk}_0, \mathrm{PK}_0)$. Once the adversary queries oracle P_0^{-1} with PK_0 which does not exist in the $T_{\mathsf{P}_0^{-1}}$-table, the simulator inserts $(*, *, P_0^{-1}(\mathrm{PK}_0), \mathrm{PK}_0)$ into P_0^{-1}-table.
5. P_1-table T_{P_1}: initially empty, consists of tuples with form of $(*, *, \mathrm{pk}_1, \mathrm{PK}_1)$. Once the adversary queries oracle P_1 with pk_1 which does not exist in T_{P_1}, the simulator inserts $(*, *, \mathrm{pk}_1, P_1(\mathrm{pk}_1))$ into T_{P_1}-table.
6. P_1^{-1}-table $T_{\mathsf{P}_1^{-1}}$: initially empty, consists of tuples with form of $(*, *, \mathrm{pk}_1, \mathrm{PK}_1)$. Once the adversary queries oracle P_1^{-1} with PK_1 which is not in $T_{\mathsf{P}_1^{-1}}$, the simulator inserts $(*, *, P_1^{-1}(\mathrm{PK}_1), \mathrm{PK}_1)$ into $T_{\mathsf{P}_1^{-1}}$-table.
7. $\mathsf{I.symKG}$-table T_{symKG}: initially empty, consists of tuples with form of $(*, \mathrm{sk}, \mathrm{pk}, *)$. Once the adversary queries $\mathsf{I.symKG}$ with sk which does not exist in T_{symKG}-table, the simulator inserts $(*, \mathrm{sk}, \mathsf{I.symKG}(\mathrm{sk}), *)$ into T_{symKG}-table.
8. $\mathsf{I.symSHK}$-table T_{symSHK}: initially empty, consists of tuples with form of $(*, \mathrm{sk}_0, \mathrm{pk}_1, K)$ or $(*, \mathrm{sk}_1, \mathrm{pk}_0, K)$. Once the adversary queries $\mathsf{I.symSHK}$ with $(\mathrm{sk}_0, \mathrm{pk}_1)$ which does not exist in the symSHK-table, the simulator inserts $(\mathrm{sk}_0, \mathrm{pk}_1, \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1))$ into T_{symSHK}-table.

In Game 0, all the queries are responded by the real oracles, and these tables are just keeping track of information related to the queries.

Claim 1. Game Real $\approx$ Game 0.

Proof. The only difference between Game Real and Game 0 is that, in Game 0 the simulator additionally maintains several tables that are hidden from the adversary, hence the adversary's views in the two games are identical, namely,

$$\Pr[\text{Game Real} = 1] = \Pr[\text{Game 0} = 1]$$

Simulator $\mathcal{S}_1$

$\underline{\mathcal{S}_1^{H_0}(\mathrm{SK}_0)}$:

Case 1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
  return sk_0;
Case 2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $\exists(*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0} \cup T_{P_0^{-1}}$ s.t. $\mathrm{PK}_0 = \Pi.\mathsf{asyKG_0}(\mathrm{SK}_0)$,
  return sk_0;
Case 3 : otherwise, $T_{H_0} = T_{H_0} \cup (\mathrm{SK}_0, H_0(\mathrm{SK}_0), *, \Pi.\mathsf{asyKG_0}(\mathrm{SK}_0))$,
  return $H_0(\mathrm{SK}_0)$.

$\underline{\mathcal{S}_1^{H_1}(\mathrm{SK}_1)}$:

Case 1 : if $\exists(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$,

return sk_1;
Case 2 : if $\exists(*, \mathrm{sk}_1, \mathrm{pk}_1, *) \in T_{\mathrm{symKG}}$ and $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1} \cup T_{P_1^{-1}}$ s.t. $\mathrm{PK}_1 = \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$,
return sk_1;
Case 3: otherwise, $T_{H_1} = T_{H_1} \cup (\mathrm{SK}_1, H_1(\mathrm{SK}_1), *, \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$,
return $H_1(\mathrm{SK}_1)$.

$\underline{\mathcal{S}_1^{P_0}(\mathrm{pk}_0)}$:
Case 1 : if $\exists(*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0} \cup T_{P_0^{-1}}$,
return PK_0;
Case 2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
return PK_0;
Case 3 : otherwise, $T_{P_0} = T_{P_0} \cup (*, *, \mathrm{pk}_0, P_0(\mathrm{pk}_0))$;
return $P_0(\mathrm{pk}_0)$.

$\underline{\mathcal{S}_1^{P_0^{-1}}(\mathrm{PK}_0)}$:
Case 1 : if $\exists(*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0} \cup T_{P_0^{-1}}$,
return pk_0;
Case 2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
return pk_0;
Case 3 : otherwise, $T_{P_0^{-1}} = T_{P_0^{-1}} \cup (*, *, \mathrm{pk}_0, P_0^{-1}(\mathrm{pk}_0))$;
return $P_0^{-1}(\mathrm{pk}_0)$.

$\underline{\mathcal{S}_1^{P_1}(\mathrm{pk}_1)}$:
Case 1 : if $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1} \cup T_{P_1^{-1}}$,
return PK_1;
Case 2 : if $\exists(*, \mathrm{sk}_1, \mathrm{pk}_1, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$,
return PK_1;
Case 3 : otherwise, $T_{P_1} = T_{P_1} \cup (*, *, \mathrm{pk}_1, P_1(\mathrm{PK}_1))$;
return $P_1(\mathrm{PK}_1)$.

$\underline{\mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)}$:
Case 1 : if $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1} \cup T_{P_1^{-1}}$,
return pk_1;
Case 2 : if $\exists(*, \mathrm{sk}_1, \mathrm{pk}_1, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$,
return pk_1;
Case 3 : otherwise, $T_{P_1^{-1}} = T_{P_1^{-1}} \cup (*, *, \mathrm{pk}_1, P_1^{-1}(\mathrm{PK}_1))$;
return $P_1^{-1}(\mathrm{PK}_1))$.

$\underline{\mathcal{S}_1^{\mathrm{symKG}}(\mathrm{sk})}$:
Case 1 : if $\exists(*, \mathrm{sk}, \mathrm{pk}, *) \in T_{\mathrm{symKG}}$,
return pk;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{sk}, *, \mathrm{PK}_0) \in T_{H_0}$,
Subcase 2.1 : if $\exists(*, *, \mathrm{pk}, \mathrm{PK}_0) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}, \mathrm{PK}_0) \in T_{P_0^{-1}}$,
return pk;
Subcase 2.2 : otherwise, $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$; $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (*, \mathrm{sk}, \mathrm{pk}, *)$,
return pk;

Case 3: if $\exists(\mathrm{SK}_1, \mathrm{sk}, *, \mathrm{PK}_1) \in T_{H_1}$,
Subcase 3.1 : if $\exists(\mathrm{SK}_1, *, \mathrm{pk}, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
return pk;
Subcase 3.2 : otherwise, $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$; $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (*, \mathrm{sk}, \mathrm{pk}, *)$,
return pk;
Case 4: otherwise, $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$; $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (*, \mathrm{sk}, \mathrm{pk}, *)$,
return pk.

//Fact: There exists difference between Subcase 2.2, Subcase 3.2 and Case 4.

//In Subcase 2.2 the adversary $\mathcal{D}$ had queried SK_0 and thus knows SK_0;

//In Subcase 3.2 the adversary $\mathcal{D}$ had queried SK_1 and thus knows SK_1;

//however, in Case 4 $\mathcal{D}$ knows neither SK_0 nor SK_1.

$\underline{\mathcal{S}_1^{\mathrm{symSHK}}(\mathrm{sk}_0, \mathrm{pk}_1)}$:

Case 1: if $\exists(\mathrm{sk}_0, \mathrm{pk}_1, K) \in T_{\mathsf{symSHK}}$,
return K;
Case 2: if $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
Subcase 2.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 2.2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$,
$K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;
Subcase 2.3 : otherwise, $K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;

//Fact: there exists difference between Subcase 2.2 and Subcase 2.3,

//in Subcase 2.2, $\mathcal{D}$ once queries sk_0 to $\mathsf{I.symKG}$ and knows pk_0;

//in Subcase 2.3, $\mathcal{D}$ never queries sk_0 and knows nothing else about sk_0.

Case 3: if $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}}$,
Subcase 3.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,
query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 3.2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$,
$K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;
Subcase 3.3 : otherwise, $K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;

//Fact: there exists difference between Subcase 3.2 and Subcase 3.3,

//in Subcase 3.2, $\mathcal{D}$ once queries sk_0 to $\mathsf{I.symKG}$ and knows pk_0;

//in Subcase 3.3, $\mathcal{D}$ never queries sk_0 and knows nothing else about sk_0.

Case 4: if $\nexists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0} \cup T_{P_1}$ or $\nexists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}} \cup T_{P_1^{-1}}$,
Subcase 4.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_0;
return K_0;
Subcase 4.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_1;

return K_1;

Subcase 4.3 : if $\exists(*, \text{sk}_0, \text{pk}_0, *) \in T_{\text{symKG}}$,

$K = \mathsf{I.symSHK}(\text{sk}_0, \text{pk}_1)$;

return K;

Subcase 4.4 : otherwise, $K = \mathsf{I.symSHK}(\text{sk}_0, \text{pk}_1)$;

return K.

//Fact: there exists difference between Subcase 4.3 and Subcase 4.4,

//in Subcase 4.3, $\mathcal{D}$ once queries sk_0 *to* I.symKG *and knows* pk_0*;*

//in Subcase 4.4, $\mathcal{D}$ never queries sk_0 *and knows nothing about* sk_0*.*

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries at adversarial interfaces. Specifically, the simulator, denoted by $\mathcal{S}_1$, responds to the oracles as above.

Compared to Game 0, in Game 1 the simulator keeps a longer table, and for part of the queries, the simulator responds to them in an alternative way, only using the tables and the honest interfaces. Moreover, in Game 1, the tuples stored in the tables are consistent with the response by the real oracles to the adversary's queries.

Claim 2. Game 0 $\approx$ Game 1.

Proof. The difference between Game 0 and Game 1 is that, in Game 1, the simulator maintains longer tables than in Game 0 and the simulator responds to part of the queries at the adversarial interfaces by using those tables and calling the honest interfaces. Moreover, the items stored in those tables are always consistent with the real oracles $H_0, H_1, P_0, P_1, P_0^{-1}, P_1^{-1}, \mathsf{I.symKG}, \mathsf{I.symSHK}$ at adversarial interfaces. Hence, for queries at adversarial interfaces, the responses by either the real oracles $H_0, H_1, P_0, P_1, P_0^{-1}, P_1^{-1}, \mathsf{I.symKG}, \mathsf{I.symSHK}$ (Game 0) or by real oracles plus honest interfaces and tables (Game 1) are identical, which implies

$$\Pr[\text{Game } 0 = 1] = \Pr[\text{Game } 1 = 1]$$

Hence, in either game, the response of any query is identical, which refers to that the view in Game 1 is identical to the one in Game 0. However, the simulator can only answer part of the queries by tables and honest interfaces, and for the rest it has to call the real oracles. Thus, in the following hybrid games, we will illustrate additional alternative ways (not calling the real oracles) to respond to the rest queries, without changing the view significantly.

Game 2. This game is identical to Game 1, except for responding to P_0 queries. In the following description of simulator, we only show the changes in the behaviors of simulator. The simulator in Game 2 responds to a query pk_0 at P_0 as follows:

Simulator $\mathcal{S}_2$

$\underline{\mathcal{S}_2^{P_0}(\mathrm{pk}_0)}$:
Case 1: if $\exists(*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0} \cup T_{P_0^{-1}}$,
return PK_0;
Case 2: if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
return PK_0;
Case 3: otherwise, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$; $T_{P_0} = T_{P_0} \cup (\tilde{\mathrm{SK}}_0, *, \mathrm{pk}_0, \tilde{\mathrm{PK}}_0)$ with $\tilde{\mathrm{PK}}_0 = \Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0)$;
return $\tilde{\mathrm{PK}}_0$.

The only difference between Game 1 and Game 2 occurs in the Case 3 where pk_0 never appears in $T_{P_0} \cup T_{P_0^{-1}} \cup T_{\mathrm{symKG}}$. In Game 1, the simulator responds to a new query pk_0 to P_0 with $P_0(\mathrm{pk}_0)$; while in Game 2, the simulator responds with $\tilde{\mathrm{PK}}_0 = \Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0)$ for a randomly sampled secret key $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$, and adds $\tilde{\mathrm{SK}}_0$ to the first item of the entry $(*, *, \mathrm{pk}_0, \tilde{\mathrm{PK}}_0)$ in table T_{P_0}. Due to definition, the only case that the adversary queries P_0 with pk_0 is when the adversary $\mathcal{D}$ knows nothing of $P_0(\mathrm{pk}_0)$. Therefore, from the adversary $\mathcal{D}$'s view, $P_0(\mathrm{pk}_0)$ is uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, $\tilde{\mathrm{PK}}_0$ is also uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, thus $\mathcal{D}$'s view in Game 2 is insidtinguishable from the view in Game 1 with high probability, except for the following bad event Bad_2.

Bad Event Bad_2**.** There exists a tuple $(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$ and a tuple $(\tilde{\mathrm{SK}}_0, *, \mathrm{pk}_0, \tilde{\mathrm{PK}}_0) \in T_{P_0}$, and $\mathcal{S}_2^{\mathrm{symKG}}(\mathrm{sk}_0)$ returns a uniformly sampled random string $\tilde{\mathrm{pk}}$ in $\{0,1\}^{n_2(\lambda)}$ s.t. $\tilde{\mathrm{pk}} = \mathrm{pk}_0$ and $\mathrm{PK}_0 \neq \tilde{\mathrm{PK}}_0$.

The Bad_2 event occurs with negligible probability, since $\tilde{\mathrm{pk}}$ is uniformly distributed and it is the same with pk_0 with probability $\frac{q}{2^{\ell_2(\lambda)}}$, which is negligible.

Claim 3. Game 1 $\approx$ Game 2.

We note that, in order not to be distinguished by the adversary, the simulator's responses at adversarial interfaces should satisfy the consistency conditions below:

1. There exists no two $\mathrm{sk}_0, \mathrm{sk}_1$ ($\mathrm{sk}_0 \neq \mathrm{sk}_1$) such that $\mathcal{S}^{\mathsf{symKG}}(\mathrm{sk}_0) = \mathcal{S}^{\mathsf{symKG}}(\mathrm{sk}_1)$;
2. $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0) = \mathcal{S}^{P_0}(\mathcal{S}^{\mathsf{symKG}}(\mathcal{S}^{H_0}(\mathrm{SK}_0)))$;
3. $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1) = \mathcal{S}^{P_1}(\mathcal{S}^{\mathsf{symKG}}(\mathcal{S}^{H_1}(\mathrm{SK}_1)))$;
4. $\Pi.\mathsf{asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1) = \mathcal{S}^{\mathsf{symSHK}}(\mathcal{S}^{H_0}(\mathrm{SK}_0), \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1))$;
5. $\Pi.\mathsf{asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0) = \mathcal{S}^{\mathsf{symSHK}}(\mathcal{S}^{H_1}(\mathrm{SK}_1), \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0))$;
6. $\mathcal{S}^{\mathsf{symSHK}}(\mathcal{S}^{H_0}(\mathrm{SK}_0), \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)) = \mathcal{S}^{\mathsf{symSHK}}(\mathcal{S}^{H_1}(\mathrm{SK}_1), \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0))$ if and only if $\mathrm{PK}_0 = \Pi.\mathsf{asyKG}_0(\mathrm{SK}_0)$ and $\mathrm{PK}_1 = \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$.
7. $\mathcal{S}^{P_0}(\mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0)) = \mathrm{PK}_0$;
8. There exists no $(\mathrm{PK}_0 \neq \mathrm{PK}_0')$, $(\mathrm{pk}_0 \neq \mathrm{pk}_0')$ such that $\mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0) = \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0')$ or $\mathcal{S}^{P_0}(\mathrm{pk}_0) = \mathcal{S}^{P_0}(\mathrm{pk}_0')$.
9. $\mathcal{S}^{P_1}(\mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)) = \mathrm{PK}_1$;
10. There exists no $(\mathrm{PK}_1 \neq \mathrm{PK}_1')$, $(\mathrm{pk}_1 \neq \mathrm{pk}_1')$ such that $\mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1) = \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1')$ or $\mathcal{S}^{P_1}(\mathrm{pk}_1) = \mathcal{S}^{P_1}(\mathrm{pk}_1')$.

Proof. The only difference between Game 1 and Game 2 occurs in the simulating the response of a P_0 query pk_0 in the Case 3, where the PK_0 corresponding to pk_0 never appears in the previous tables. In Game 1, the simulator responds to pk_0 with $P_0(\mathrm{pk}_0)$, while in Game 2, the simulator samples a uniformly random $\tilde{\mathrm{SK}}_0$ and responds with $\Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0)$.

Since P_0 is a random permutation, the distribution of $P_0(\mathrm{pk}_0)$ should be uniformly random in $\mathsf{DomAsy}_{\mathrm{PK}_0}$, and $\Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0)$ is also uniformly random in $\mathsf{DomAsy}_{\mathrm{PK}_0}$ for a random $\tilde{\mathrm{SK}}_0$.

For the Case 3, it's trivial that the adversary had never made a query $\mathcal{S}^{\mathsf{symKG}}(\mathcal{S}^{H_0}(\mathrm{SK}_0))$ to P_0, as such a query would have resulted in a tuple which contains PK_0 being added to P_0 table. Therefore, SK_0, $\mathcal{S}^{H_0}(\mathrm{SK}_0)$ and $\mathcal{S}^{\mathsf{symKG}}(\mathcal{S}^{H_0}(\mathrm{SK}_0))$ are independent of pk_0 in the adversary's view. Besides, it's easy to check that in Game 2, the equations for consistency hold. Combining together, Game 1 and Game 2 are indistinguishable except that Bad_2 occurs. Hence,

$$|\Pr[\text{Game } 1 = 1] - \Pr[\text{Game } 2 = 1]| \leq q \cdot \Pr[\mathsf{Bad}_2] \leq \mathsf{negl}(\lambda)$$

Game 3. This game is identical to Game 2, except for responding to P_0^{-1} queries. The simulator responds to a query PK_0 at P_0^{-1} as follows:

Simulator $\mathcal{S}_3$

$\mathcal{S}_3^{P_0^{-1}}(\mathrm{PK}_0)$:

Case 1: if $\exists(\mathrm{SK}_0, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0} \cup (*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0^{-1}}$,
 return pk_0;
Case 2: if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
 return pk_0;
Case 3: $\tilde{\mathrm{pk}}_0 \leftarrow \mathsf{DomSym}_{\mathrm{pk}}$, $T_{P_0^{-1}} = T_{P_0^{-1}} \cup (*, *, \tilde{\mathrm{pk}}_0, \mathrm{PK}_0)$;
 return $\tilde{\mathrm{pk}}_0$.

The only difference between Game 2 and Game 3 occurs in the Case 3 where PK_0 is never queried to P_0^{-1}, besides, the corresponding SK_0 is never queried to H_0 and $H_0(\mathrm{SK}_0)$ is never queried to P_0.

For the Case 3, in Game 2, the simulator responds to P_0^{-1} query on PK_0 with $P_0^{-1}(\mathrm{PK}_0)$; while in Game 3, the simulator responds with a random string $\tilde{\mathrm{pk}}_0$. Due to definition, the only case that the adversary queries P_0^{-1} with PK_0 is when the adversary $\mathcal{D}$ knows nothing of $P_0^{-1}(\mathrm{PK}_0)$. Therefore, from the adversary $\mathcal{D}$'s view, $P_0^{-1}(\mathrm{PK}_0)$ is uniformly distributed in $\{0,1\}^{n_2(\lambda)}$, which implies $\mathcal{D}$'s view in Game 3 preserves with high probability if the simulator responds a P_0^{-1} query PK_0 with a uniformly random string $\tilde{\mathrm{pk}}_0$ in $\{0,1\}^{n_2(\lambda)}$ and stores a tuple $(*, *, \tilde{\mathrm{pk}}_0, \mathrm{PK}_0)$ in Table $T_{P_0^{-1}}$, unless the following bad event occurs.

Bad Event Bad_3**.** There exists a tuple $(\mathrm{SK}_0', \mathrm{sk}_0, *, \mathrm{PK}_0') \in T_{H_0}$ and a tuple $(*, *, \tilde{\mathrm{pk}}_0, \mathrm{PK}_0) \in T_{P_0^{-1}}$, and $\mathcal{S}_3^{\mathrm{symKG}}(\mathrm{sk}_0)$ uniformly samples a random string $\tilde{\mathrm{pk}}$ in $\{0,1\}^{n_2(\lambda)}$ s.t. $\tilde{\mathrm{pk}} = \tilde{\mathrm{pk}}_0$ and $\mathrm{PK}_0' \neq \mathrm{PK}_0$. The bad event occurs with negligible probability, since $\tilde{\mathrm{pk}}$ is uniformly distributed and its value equals $\tilde{\mathrm{pk}}_0$ with probability $\frac{q}{2^{n_2(\lambda)}}$, which is negligible.

Claim 4. Game 2 $\approx$ Game 3.

Proof. The only difference between Game 2 and Game 3 occurs in simulating the response of a P_0^{-1} query PK_0 in the Case 3, where pk_0 corresponding to PK_0 never appears in $T_{P_0} \cup T_{P_0^{-1}} \cup T_{\mathrm{symKG}}$. In Game 2, the simulator responds to a query PK_0 on P_0^{-1} with $P_0^{-1}(\mathrm{PK}_0)$, while in Game 3, the simulator samples a uniformly random string $\tilde{\mathrm{pk}}_0$.

Since P_0^{-1} is a random permutation, the distribution of $P_0^{-1}(\mathrm{PK}_0)$ should be uniformly distributed in $\mathsf{DomAsy}_{\mathrm{PK}_0}$. Note that $\tilde{\mathrm{pk}}_0$ is also uniformly distributed in $\mathsf{DomAsy}_{\mathrm{PK}_0}$. Since the adversary has no ability to learn $P_0^{-1}(\mathrm{PK}_0)$ in the Case 3, thus the simulator can answer a P_0^{-1} query PK_0 with a random $\tilde{\mathrm{pk}}_0$, such that the adversary'e view in Game 2 and Game 3 are indistinguishable with high probability except that the event Bad_3 occurs. Besides, in Game 3, the consistency equations holds trivially. Hence,

$$|\Pr[\text{Game } 2 = 1] - \Pr[\text{Game } 3 = 1]| \leq q \cdot \Pr[\mathsf{Bad}_3] \leq \mathsf{negl}(\lambda)$$

Game 4. This game is identical to Game 3, except for responding to P_1 queries. The simulator responds to a query pk_1 at P_1 in the same way as in Game 3, except when the response of $P_1(\mathrm{pk}_1)$ cannot be inferred from the entries of tables maintained by the simulator. In the exception case, the simulator randomly sample $\tilde{\mathrm{SK}}_1 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_1}$; add $(\tilde{\mathrm{SK}}_1, *, \mathrm{pk}_1, \tilde{\mathrm{PK}}_1)$ to the table T_{P_1}, compute $\tilde{\mathrm{PK}}_1 = \Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_1)$; and responds the query pk_1 at P_1 with $\tilde{\mathrm{PK}}_1$.

The only difference between Game 3 and Game 4 occurs in the exception case, where pk_1 never appears in $T_{P_1} \cup T_{P_1^{-1}} \cup T_{\mathrm{symKG}}$.

In Game 3, the simulator responds to P_1 query on pk_1 with $P_1(\mathrm{pk}_1)$; while in Game 4, the simulator responds with a string $\tilde{\mathrm{PK}}_1 = \Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_1)$ for a randomly sampled string $\tilde{\mathrm{SK}}_1$. Due to definition, the only case that the adversary queries P_1 with pk_1 is when the adversary $\mathcal{D}$ knows nothing of $P_1(\mathrm{pk}_1)$. Therefore, from the adversary $\mathcal{D}$'s view, $P_1(\mathrm{pk}_1)$ is uniformly distributed in $\{0,1\}^{n_2(\lambda)}$, which implies $\mathcal{D}$'s view in Game 4 preserves with high probability if the simulator responds a P_1 query pk_1 with a string $\tilde{\mathrm{PK}}_1$ that is also uniformly distributed in $\{0,1\}^{n_2(\lambda)}$ and adds $\tilde{\mathrm{SK}}_1$ in an entry $(\tilde{\mathrm{SK}}_1, *, \mathrm{pk}_1, \tilde{\mathrm{PK}}_1)$ of Table T_{P_1}.

Bad Event Bad_4**.** There exists a tuple $(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$ and a tuple $(\tilde{\mathrm{SK}}_1, *, \mathrm{pk}_1, \tilde{\mathrm{PK}}_1) \in T_{P_1}$, and $\mathcal{S}_4^{\mathrm{symKG}}(\mathrm{sk}_1)$ returns a uniformly sampled string $\tilde{\mathrm{pk}}$ in $\{0,1\}^{n_2(\lambda)}$ s.t. $\tilde{\mathrm{pk}} = \mathrm{pk}_1$ and $\mathrm{PK}_1 \neq \tilde{\mathrm{PK}}_1$.

The bad event Bad_4 occurs with negligible probability, since $\tilde{\mathrm{pk}}$ is uniformly distributed and its value equals pk_1 with probability $\frac{q}{2^{n_2(\lambda)}}$, which is negligible.

Claim 5. Game 3 $\approx$ Game 4.

The only difference between Game 3 and Game 4 occurs in the simulating the response of a P_1 query pk_1 in the Case 3, where the PK_1 corresponding to pk_0 never appears in the previous tables. In Game 3, the simulator responds to pk_1 with $P_1(\mathrm{pk}_1)$, while in Game 4, the simulator samples a uniformly random $\tilde{\mathrm{SK}}_1$ and responds with $\Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_1)$. Similarly to the analysis for Claim 3, Game 3 and Game 4 are indistinguishable except that Bad_4 occurs, hence

$$|\Pr[\text{Game } 3 = 1] - \Pr[\text{Game } 4 = 1]| \leq q \cdot \Pr[\mathsf{Bad}_4] \leq \mathsf{negl}(\lambda)$$

Game 5. This game is identical to Game 4, except for responding to P_1^{-1} queries. The simulator responds to a P_1^{-1} query PK_1 as follows:

Simulator $\mathcal{S}_5$

$\mathcal{S}_5^{P_1^{-1}}(\mathrm{PK}_1)$:

Case 1: if $\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1} \cup \exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
  return pk_1;
Case 2: if $\exists(*, \mathrm{sk}_1, \mathrm{pk}_1, *) \in T_{\mathrm{symKG}}$ and $\exists(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$,
  return pk_1;
Case 3: otherwise, $\tilde{\mathrm{pk}}_1 \leftarrow \mathsf{DomSym}_{\mathrm{pk}}$, $T_{P_1^{-1}} = T_{P_1^{-1}} \cup (*, *, \tilde{\mathrm{pk}}_1, \mathrm{PK}_1)$;
  return $\tilde{\mathrm{pk}}_1$.

The only difference between Game 4 and Game 5 occurs in the case where PK_1 never appears in $T_{P_0} \cup T_{P_0^{-1}} \cup T_{H_1}$. In Game 4, the simulator responds to P_1^{-1} query on PK_1 with $P_1^{-1}(\mathrm{PK}_1)$; while in Game 5, the simulator responds with a random string $\tilde{\mathrm{pk}}_1$. Due to definition, the only case that the adversary queries P_1^{-1} with PK_1 is when the adversary $\mathcal{D}$ knows nothing of $P_1^{-1}(\mathrm{PK}_1)$. Therefore, from the adversary $\mathcal{D}$'s view, $P_1^{-1}(\mathrm{PK}_1)$ is uniformly distributed in $\{0,1\}^{n_2(\lambda)}$, which implies $\mathcal{D}$'s view in Game 7 preserves with high probability if the simulator responds a P_1^{-1} query PK_1 with a uniformly random string $\tilde{\mathrm{pk}}_1$ in $\{0,1\}^{n_2(\lambda)}$ and stores a tuple $(*, *, \tilde{\mathrm{pk}}_1, \mathrm{PK}_1)$ in Table $T_{P_1^{-1}}$, unless the following bad event occurs.

Bad Event Bad_5. There exists a tuple $(\mathrm{SK}_1', \mathrm{sk}_1, *, \mathrm{PK}_1') \in T_{H_1}$ and a tuple $(*, *, \tilde{\mathrm{pk}}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$, and $\mathcal{S}_8^{\mathrm{symKG}}(\mathrm{sk}_1)$ returns a uniformly sampled random string $\tilde{\mathrm{pk}}$ in $\{0,1\}^{n_2(\lambda)}$ s.t. $\tilde{\mathrm{pk}} = \tilde{\mathrm{pk}}_1$ and $\mathrm{PK}_1' \neq \mathrm{PK}_1$. The bad event Bad_5 occurs with negligible probability, since $\tilde{\mathrm{pk}}$ is uniformly distributed and its value equals $\tilde{\mathrm{pk}}_1$ with probability $\frac{q}{2^{n_2(\lambda)}}$, which is negligible.

Claim 6. Game 4 $\approx$ Game 5. Similarly to the analysis in Claim 4, Game 4 and Game 5 are indistinguishable except that the event Bad_5 occurs. Hence,

$$|\Pr[\text{Game } 4 = 1] - \Pr[\text{Game } 5 = 1]| \leq q \cdot \Pr[\mathsf{Bad}_5] \leq \mathsf{negl}(\lambda)$$

Game 6. This game is identical to Game 5, except for responding to H_0 queries. In Game 6, the simulator responds to a query SK_0 at H_0 as follows:

Simulator $\mathcal{S}_6$

$\underline{\mathcal{S}_6^{H_0}(\mathrm{SK}_0)}$:

Case 1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
  return sk_0;
Case 2 : if $\exists(*, \mathrm{sk}_0, \mathrm{pk}_0, *) \in T_{\mathrm{symKG}}$ and $(\exists(\mathrm{SK}_0, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{P_0^{-1}}$, s.t. $\mathrm{PK}_0 = \Pi.\mathsf{asyKG}_0(\mathrm{SK}_0))$,
  return sk_0;
Case 3 : otherwise, $\mathrm{sk}_0 \leftarrow \mathsf{DomSym}_{\mathrm{sk}}$; $T_{H_0} = T_{H_0} \cup (\mathrm{SK}_0, \mathrm{sk}_0, *, \Pi.\mathsf{asyKG}_0(\mathrm{SK}_0))$;
  return sk_0.

The only difference between Game 5 and Game 6 occurs in the Case 3 where SK_0 never appears in T_{H_0}, and it never occurs that sk_0 appears in table T_{symKG} and $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0)$ appears in $\mathsf{P}_1 \cup \mathsf{P}_1^{-1}$ table simultaneously.

In Game 5, the simulator responds a query SK_0 at H_0 with $H_0(\mathrm{SK}_0)$, while in Game 6, the simulator responds with a random string sk_0. Due to definition, the only case that the simulator queries H_0 with SK_0 is when the adversary $\mathcal{D}$ knows nothing of $H_0(\mathrm{SK}_0)$, although the adversary might know $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0)$. Therefore, from the adversary $\mathcal{D}$'s view, $H_0(\mathrm{SK}_0)$ is uniformly distributed in $\{0,1\}^{n_1(\lambda)}$, which implies $\mathcal{D}$'s view in Game 6 is statistically indistinguishable from its view in Game 5.

Claim 7. Game 5 $\approx$ Game 6.

Proof. Recalling that the only difference between Game 5 and Game 6 occurs in simulating the response of a H_0 query SK_0 in the Case 3, where the SK_0. In Game 5, the simulator responds to a H_0 query SK_0 with $H_0(\mathrm{SK}_0)$ while in Game 6, the simulator replaces it with a random string sk_0 in $\mathsf{DomAsy}_{\mathrm{sk}}$.

To prove the indistinguishability, we first formalize the adversary's view in Game 5. By definition, in Game 5,

- The simulator responds to a H_0 query SK_0 with $H_0(\mathrm{SK}_0)$;
- The simulator responds to a H_1 query SK_1 with $H_1(\mathrm{SK}_1)$;
- The simulator responds to a P_0 query PK_0 with $\mathcal{S}^{P_0}(\mathrm{PK}_0)$;
- The simulator responds to a P_0^{-1} query pk_0 with $\mathcal{S}^{P_0^{-1}}(\mathrm{pk}_0)$;
- The simulator responds to a P_1 query PK_1 with $\mathcal{S}^{P_1}(\mathrm{PK}_1)$;
- The simulator responds to a P_1^{-1} query pk_1 with $\mathcal{S}^{P_1^{-1}}(\mathrm{pk}_1)$;
- The simulator responds to a symKG query sk with $\mathsf{I.symKG}(\mathrm{sk})$;
- The simulator responds to a symSHK query $(\mathrm{sk}_b, \mathrm{pk}_{1-b})$ for $b \in \{0,1\}$ with $\mathsf{I.symSHK}(\mathrm{sk}_b, \mathrm{pk}_{1-b})$;

Hence in adversary's view, under the consistency conditions, the responses of H_0 and H_1 are independent and random strings; the responses of $\mathcal{S}^{P_0}, \mathcal{S}^{P_0^{-1}}$

are indistinguishable from those of random permutations, as are the same for $(\mathcal{S}^{P_1}, \mathcal{S}^{P_0^{-1}})$, and the behavior of $(\mathcal{S}^{P_0}, \mathcal{S}^{P_0^{-1}})$ are independent of $(\mathcal{S}^{P_1}, \mathcal{S}^{P_0^{-1}})$ (; $\mathsf{I.symKG}$ a random injection, and $\mathsf{I.symSHK}$ is a random injection with shared key property.

Next we see the view on the adversarial interfaces in Game 6. For any H_0 query SK_0, the simulator's response sk_0 is uniformly sampled and has the same distribution with $H_0(\mathrm{SK}_0)$, thus the responses for H_0 queries are indistinguishable in Game 5 and Game 6. We note that for $H_1, P_0, P_1, P_0^{-1}, P_1^{-1}, \mathsf{I.symKG}, \mathsf{I.symSHK}$ queries, the responses are identical in Game 5 and Game 6, without being influenced by the replacement of $\mathcal{S}^{H_0}(\mathrm{SK}_0)$ with $H_0(\mathrm{SK}_0)$.

Let $\mathsf{Collide}_6$ denote the event that $\mathcal{S}^{H_0}$ responds a query SK_0 with a string sk_0, while $(\mathrm{SK}_0', \mathrm{sk}_0, *, *)$ $(\mathrm{SK}_0' \neq \mathrm{SK}_0)$ already exists in the tables maintained by the simulator. Namely, there is a collision in the responses of H_0 queries. In fact, the simulator's response is uniformly sampled from $\mathsf{DomSym}_{\mathrm{sk}}$, the probability of $\mathsf{Collide}_6$ occurs is bounded by $\frac{q}{|\mathsf{DomSym}_{\mathrm{sk}}|}$.

Next we prove that, with high probability, the consistency conditions in Game 6 hold.

First Equation. As $\mathsf{I.symKG}$ is a random injection, and the responses of $\mathcal{S}^{\mathsf{symKG}}$ are consistent with those of $\mathsf{I.symKG}$ (namely, the response of $\mathcal{S}^{\mathsf{symKG}}(\mathrm{sk})$ is indistinguishable from that of $\mathsf{I.symKG}(\mathrm{sk})$ for any $\mathrm{sk} \in \mathsf{DomSym}_{\mathrm{sk}}$, and both of them are consistent with the responses at honest interfaces), thus this equation holds trivially with high probability.

Second Equation. As $\mathsf{I.symKG}$ is a random injection, P_0 is a random permutation, and the responses of $\mathcal{S}^{H_0}$, $\mathcal{S}^{P_0}$ and $\mathcal{S}^{\mathsf{symKG}}$ are consistent with H_0, P_0 and $\mathsf{I.symKG}$, respectively; hence the equation holds.

Third Equation. As $\mathsf{I.symKG}$ is a random injection, P_1 is a random permutation, meanwhile, the responses of $\mathcal{S}^{P_1}$, $\mathcal{S}^{\mathsf{symKG}}$ and $\mathcal{S}^{H_1}$ are consistent with those of P_1, $\mathsf{I.symKG}$ and H_1, respectively. Therefore, by definition the equation holds trivially.

Fourth Equation. As $\mathsf{I.symSHK}$ is a random injection, P_1^{-1} is a random permutation, meanwhile, the responses of $\mathcal{S}^{\mathrm{symSHK}}$, $\mathcal{S}^{P_1^{-1}}$ are consistent with $\mathsf{I.symSHK}$ and P_1^{-1}, respectively, hence the equation holds.

Fifth Equation. As $\mathsf{I.symSHK}$ is a random injection, P_0^{-1} is a random permutation, meanwhile, the responses of $\mathcal{S}^{\mathrm{symSHK}}$, $\mathcal{S}^{P_0^{-1}}$ $\mathcal{S}^{H_1}$ are consistent with those of $\mathsf{I.symSHK}$, P_0^{-1} and H_1, respectively. By definition, the equation holds.

Sixth Equation. Under the condition that the Fourth Equation and the Fifth Equation holds and the shared key property of $\mathsf{I.symSHK}$ holds, this equation holds.

Seventh Equation. This equation holds trivially since the responses of P_0 and P_0^{-1} are consistent.

Eighth Equation. This equation holds trivially since the simulator's responses of P_0 and P_0^{-1} are indistinguishable from that of random permutations except with negligible probability, bounded by $\frac{q}{|\mathsf{DomSym}_{\mathrm{pk}}|}$.

Ninth Equation. This equation holds trivially since the responses of P_1 and P_1^{-1} are consistent.

Tenth Equation. This equation holds trivially since the simulated responses of P_1 and P_1^{-1} are indistinguishable from the responses of random permutations except with negligible probability, which is bounded by $\frac{q}{|\mathsf{DomSym}_{\mathrm{pk}}|}$.

According to the analysis above, the adversary's views in Game 5 and Game 6 are indistinguishable, which refers to

$$|\Pr[\text{Game } 5 = 1] - \Pr[\text{Game } 6 = 1]| \leq q \cdot \Pr[\mathsf{Collide}_6] \leq \mathsf{negl}(\lambda)$$

Game 7. This game is identical to Game 6, except for responding to H_1 queries. The simulator responds to a query SK_1 at H_1 as follows:

Simulator $\mathcal{S}_7$

$\underline{\mathcal{S}_7^{H_1}(\mathrm{SK}_1)}$:
Case 1 : if $\exists(\mathrm{SK}_1, \mathrm{sk}_1, *, \mathrm{PK}_1) \in T_{H_1}$,
return sk_1;
Case 2 : if $\exists(*, \mathrm{sk}_1, \mathrm{pk}_1, *) \in T_{\mathrm{symKG}}$ and ($\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$ s.t. $\mathrm{PK}_1 = \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$,
return sk_1;
Case 3 : otherwise, $\mathrm{sk}_1 \leftarrow \mathsf{DomSym}_{\mathrm{sk}}$; $T_{H_1} = T_{H_1} \cup (\mathrm{SK}_1, \mathrm{sk}_1, *, \Pi.\mathsf{asyKG}_1(\mathrm{SK}_1))$, return sk_1.

The only difference between Game 6 and Game 7 occurs in the Case 3 where SK_1 never appears in T_{H_1}, and it never occurs that sk_1 appears in table T_{symKG} and $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$ appears in $\mathsf{P}_1 \cup \mathsf{P}_1^{-1}$ table simultaneously. In Game 6, the simulator responds to H_1 query on SK_1 with $H_1(\mathrm{SK}_1)$; while in Game 7, the simulator responds with a random string sk_1. Due to definition, the only case that the simulator queries H_1 with SK_1 is when the adversary $\mathcal{D}$ knows nothing of $H_1(\mathrm{SK}_1)$, although the adversary might know $\Pi.\mathrm{asyKG}_1(\mathrm{SK}_1)$. Therefore, from the adversary $\mathcal{D}$'s view, $H_1(\mathrm{SK}_1)$ is uniformly distributed in $\{0,1\}^{n_1(\lambda)}$, which implies that $\mathcal{D}$'s view in Game 7 is statistically indistinguishable from its view in Game 6.

Claim 8. Game 6 $\approx$ Game 7.

Proof. The analysis in this claim is very similar to the Claim 7 (Game 5 $\approx$ Game 6), hence

$$|\Pr[\text{Game } 6 = 1] - \Pr[\text{Game } 7 = 1]| \leq |\Pr[\text{Game } 5 = 1] - \Pr[\text{Game } 6 = 1]| \leq \mathsf{negl}(\lambda)$$

Game 8. This game is identical to Game 7, except for responding to symKG queries. Remark that responses to symSHK queries change according to that of symKG queries. The simulator responds to a symKG query sk as in the box of Simulator $\mathcal{S}_8$.

The only difference between Game 7 and Game 8 occurs in the Subcase 2.2, Subcase 3.2 and Case 4. For the Subcase 2.2, in Game 7, the simulator responds with $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$ and inserts $(*, \mathrm{sk}, \mathrm{pk}, *)$ in T_{symKG}; while in Game 8, the simulator responds with $\mathrm{pk} = \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0)$ and inserts $(\mathrm{SK}_0, \mathrm{sk}, \mathrm{pk}, \mathrm{PK}_0)$ in T_{symKG}. In Subcase 2.2, there exists a tuple $(\mathrm{SK}_0, \mathrm{sk}, *, \mathrm{PK}_0)$ in T_{H_0}, sk is never queried to $\mathsf{I.symKG}$ and PK_0 is never queried to P_0^{-1}, which implies that pk is uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, and the distributions of $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$ and $\mathrm{pk} = \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0)$ are identical. Besides, inserting longer tables for T_{symKG} will not be detected by the adversary.

Simulator $\mathcal{S}_8$

$\mathcal{S}_8^{\mathrm{symKG}}(\mathrm{sk})$:

Case 1 : if $\exists(*, \mathrm{sk}, \mathrm{pk}, *) \in T_{\mathrm{symKG}}$,
  return pk;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{sk}, *, \mathrm{PK}_0) \in T_{H_0}$,
  Subcase 2.1 : if $\exists(\mathrm{SK}_0, *, \mathrm{pk}, \mathrm{PK}_0) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}, \mathrm{PK}_0) \in T_{P_0^{-1}}$,
   return pk;
  Subcase 2.2 : otherwise, run $\mathrm{pk} \leftarrow \mathcal{S}^{P_0^{-1}}(\mathrm{PK}_0)$; $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (\mathrm{SK}_0, \mathrm{sk}, \mathrm{pk}, \mathrm{PK}_0)$,
   return pk;
Case 3: if $\exists(\mathrm{SK}_1, \mathrm{sk}, *, \mathrm{PK}_1) \in T_{H_1}$,
  Subcase 3.1 : if $\exists(\mathrm{SK}_1, *, \mathrm{pk}, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
   return pk;
  Subcase 3.2 : otherwise, run $\mathrm{pk} \leftarrow \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)$, $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (\mathrm{SK}_1, \mathrm{sk}, \mathrm{pk}, \mathrm{PK}_1)$,
   return pk;
Case 4: otherwise, $\tilde{\mathrm{pk}} \leftarrow \mathsf{DomSym}_{\mathrm{pk}}$, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$, $\tilde{\mathrm{SK}}_1 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_1}$;
  query $\mathsf{I.asyKG}_0$ with $\tilde{\mathrm{SK}}_0$, obtain $\tilde{\mathrm{PK}}_0$; query $\mathsf{I.asyKG}_1$ with $\tilde{\mathrm{SK}}_1$, obtain $\tilde{\mathrm{PK}}_1$;
$T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (\tilde{\mathrm{SK}}_0, \mathrm{sk}, \tilde{\mathrm{pk}}, \tilde{\mathrm{PK}}_0)$, $T_{\mathrm{symKG}} = T_{\mathrm{symKG}} \cup (\tilde{\mathrm{SK}}_1, \mathrm{sk}, \tilde{\mathrm{pk}}, \tilde{\mathrm{PK}}_1)$;
  return $\tilde{\mathrm{pk}}$.

//From this point, all entries in T_{symKG} are updated as the form $(\mathrm{SK}, \mathrm{sk}, \mathrm{pk}, \mathrm{PK})$.

$\mathcal{S}_8^{\mathrm{symSHK}}(\mathrm{sk}_0, \mathrm{pk}_1)$:

Case 1: if $\exists(\mathrm{sk}_0, \mathrm{pk}_1, K) \in T_{\mathsf{symSHK}}$,
  return K;
Case 2: if $\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
  Subcase 2.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
   query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
   return K;
  Subcase 2.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,

query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 2.3 : otherwise, $K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;
Case 3: if $\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}}$,
Subcase 3.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,
query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 3.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,
query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 3.3 : otherwise, $K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K;
Case 4: if $\nexists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0} \cup T_{P_1}$ or $\nexists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}} \cup T_{P_1^{-1}}$,
Subcase 4.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_0;
return K_0;
Subcase 4.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_1;
return K_1;
Subcase 4.3 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
//or run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K; return K;
Subcase 4.4 : otherwise, $K = \mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$;
return K.

For the Subcase 3.2, the analysis is similar to that of Subcase 2.2. In Game 7, the simulator responds with $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$ and inserts $(*, \mathrm{sk}, \mathrm{pk}, *)$ in T_{symKG}; in Game 8, the simulator responds with $\mathrm{pk} \leftarrow \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)$ and inserts $(\mathrm{SK}_1, \mathrm{sk}, \mathrm{pk}, \mathrm{PK}_1)$ in T_{symKG}. Since there exists a tuple $(\mathrm{SK}_1, \mathrm{sk}, *, \mathrm{PK}_1)$ in T_{H_1}, where sk is never queried to $\mathsf{I.symKG}$ and PK_1 is never queried to P_1^{-1}, we have that pk is uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, and the distributions of $\mathrm{pk} = \mathsf{I.symKG}(\mathrm{sk})$ and $\mathrm{pk} = \mathcal{S}^{P_1^{-1}}(\mathrm{PK}_1)$ are identical. Besides, inserting longer tables for T_{symKG} will not be detected by the adversary.

For the Case 4, in Game 7, the simulator responds a query to $\mathsf{I.symKG}$ with $\mathsf{I.symKG}(\mathrm{sk})$. In Game 8, the simulator responds with a randomly sampled string $\tilde{\mathrm{pk}} \leftarrow \{0,1\}^{n_2(\lambda)}$; meanwhile, the simulator samples $\tilde{\mathrm{SK}}_0$ and $\tilde{\mathrm{SK}}_0$, and implicitly set $H_0(\tilde{\mathrm{SK}}_0) = \mathrm{sk}$, $H_1(\tilde{\mathrm{SK}}_1) = \mathrm{sk}$, $P_0(\tilde{\mathrm{pk}}) = \Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0)$ and $P_1(\tilde{\mathrm{pk}}) = \Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_1)$. The reason that $P_0(\tilde{\mathrm{pk}})$, $P_1(\tilde{\mathrm{pk}})$ are set as random public keys, rather than random strings, is to keep consistency with the honest interfaces $\Pi.\mathsf{asySHK}_0$ and $\Pi.\mathsf{asySHK}_1$. Due to definition, the only case that the adversary queries $\mathsf{I.symKG}$ with sk is when the adversary $\mathcal{D}$ knows nothing of $\mathsf{I.symKG}(\mathrm{sk})$. Besides, in Case 4, $\mathcal{D}$ knows nothing about the SK_0 and SK_1 corresponding to sk, which implies that $\Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0), \Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_1)$ are also well-distributed. Beyond that, the responses of symSHK queries have minor changes in the Subcase

2.2, Subcase 3.2 and Subcase 4.3, due to the change of table T_{symKG}. Note that the changes do not influence the consistency conditions and the view of the adversary $\mathcal{D}$ in Game 8.

Therefore, from the adversary $\mathcal{D}$'s view, $\mathsf{I.symKG}(\text{sk})$ is uniformly distributed in $\{0,1\}^{n_2(\lambda)}$, and $\mathcal{D}$'s view in Game 8 are indistinguishable from its view in Game 7 with high probability.

Claim 9. Game 7 $\approx$ Game 8.

Proof.

The adversary's view in Game 8 is: the responses of H_0, H_1 queries are random and independent strings; the responses of P_0, P_0^{-1} queries and P_1, P_1^{-1} queries are independent random permutations; $\mathsf{I.symSHK}$ is a random injection with the shared key property.

The differences between Game 7 and Game 8 occurs in the Subcase 2.2, the Subcase 3.2 and the Case 4 for simulating the responses of symKG queries, as well as in the Subcase 2.2, the Subcase 3.2 and the Subcase 4.3 for simulating the responses of symSHK queries.

First, we discuss the differences in simulating the responses of symKG queries. In the Subcase 2.2, for a symKG query sk, in Game 7 the simulator responds with $\mathsf{I.symKG}(\text{sk})$ and inserts $(*, \text{sk}, \mathsf{I.symKG}(\text{sk}), *)$ in the table T_{symKG}; while in Game 8, the simulator responds with $\text{pk} = \mathcal{S}^{P_0^{-1}}(\text{PK}_0)$ and inserts $(\text{SK}_0, \text{sk}, \text{pk}, \text{PK}_0)$ in the table T_{symKG}. By the definition of $\mathcal{S}^{P_0^{-1}}$, $\text{pk} = \mathcal{S}^{P_0^{-1}}(\text{PK}_0)$ and $\mathsf{I.symKG}(H_0(\text{SK}_0))$ have the same distribution. In addition, the responses of H_0, P_0, P_0^{-1} queries are consistent in both Game 7 and Game 8 (for both Game 7 and Game 8, the consistency conditions hold). Storing longer tables will not be detected by the adversary or influence the consistency conditions. Therefore, in the Subcase 2.2, the adversary's views in Game 7 and Game 8 are indistinguishable.

In the Subcase 3.2, the adversary's views in Game 7 and Game 8 are indistinguishable, and the analysis is similar to that of Subcase 2.2.

In the Case 4, for a symKG query sk, in Game 7 the simulator responds with $\mathsf{I.symKG}(\text{sk})$; while in Game 8, the simulator responds with a random sampled $\tilde{\text{pk}} \leftarrow \mathsf{DomSym}_{\text{pk}}$, meanwhile, the simulator randomly selects $\widetilde{\text{SK}_0}$, $\widetilde{\text{SK}_1}$ from $\mathsf{DomAsy}_{\text{SK}_0}$ and $\mathsf{DomAsy}_{\text{SK}_1}$, respectively, inserts $(\widetilde{\text{SK}_0}, \text{sk}, \tilde{\text{pk}}, \Pi.\mathsf{asyKG}_0(\widetilde{\text{SK}_0}))$ and $(\widetilde{\text{SK}_1}, \text{sk}, \tilde{\text{pk}}, \Pi.\mathsf{asyKG}_1(\widetilde{\text{SK}_1}))$ in the table T_{symKG}. In addition, the simulator implicitly sets $\mathcal{S}^{H_0}(\widetilde{\text{SK}_0}) = \text{sk}$, $\mathcal{S}^{H_1}(\widetilde{\text{SK}_1}) = \text{sk}$, $\mathcal{S}^{P_0}(\text{pk}) = \Pi.\mathsf{asyKG}_0(\widetilde{\text{SK}_0})$ and $\mathcal{S}^{P_1}(\text{pk}) = \Pi.\mathsf{asyKG}_1(\widetilde{\text{SK}_1})$. Note that $\mathsf{I.symKG}(\text{sk})$ and $\tilde{\text{pk}}$ have the same distribution.

Let Bad_8 denote the event that the adversary makes one of the following queries before the symKG query sk:

- 1) query SK_0 to real oracle H_0, which responds with sk
- 2) query SK_1 to real oracle H_1, which responds with sk
- 3) query $\Pi.\mathsf{asyKG}_0(\text{SK}_0)$ to real oracle P_0^{-1}, which responds with pk

– 4) query $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$ to real oracle P_1^{-1}, which responds with pk

Hence as long as Bad_8 did not occur, in the Case 4, the responses of H_0, H_1 and $P_0, P_0^{-1}, P_1, P_1^{-1}$ queries will be consistent in Game 7 and Game 8.

In fact, SK_0 and SK_1 are hidden from the adversary, thus $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0)$ and $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$ are also random strings never revealed to the adversary, which means Bad_8 occurs(i.e., the four bad queries appears in the query sequence) with probability bounded by $\frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_1}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_1}|}$, which is negligible.

Hence as long as the corresponding SK_0, and $H_0(\mathrm{SK}_0)$, SK_1, $H_1(\mathrm{SK}_1)$, as well as $P_0^{-1}(\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0))$ and $P_1^{-1}(\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1))$ are hidden from the adversary, the responses of symKG queries are indistinguishable in Game 7 and Game 8. It's trivial that the responses in Game 8 satisfy the consistency conditions.

For symSHK queries, the difference is caused by the change of T_{symKG} table, and the responses of symSHK queries are consistent in Game 7 and Game 8 if the responses of symKG are consistent.

Combining together, the adversary's view in Game 7 and Game 8 are indistinguishable except when the event Bad_8 occurs, and we can bound probability of Bad_8 by the following equation.

$$\Pr[\mathsf{Bad}_8] \leq \frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_1}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_1}|}$$

which refers to

$$|\Pr[\text{Game } 7 = 1] - \Pr[\text{Game } 8 = 1]| \leq q \cdot \Pr[\mathsf{Bad}_8] \leq \mathsf{negl}(\lambda)$$

Game 9. This game is identical to Game 8, except for responding to symSHK queries.

Simulator $\mathcal{S}_9$

$\underline{\mathcal{S}_9^{\mathrm{symSHK}}(\mathrm{sk}_0, \mathrm{pk}_1)}$:
Case 1: if $\exists(\mathrm{sk}_0, \mathrm{pk}_1, K) \in T_{\mathsf{symSHK}}$,
  return K;
Case 2: if $\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_1^{-1}}$,
  Subcase 2.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
   query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
   return K;
  Subcase 2.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,
   query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
   return K;
  Subcase 2.3 : otherwise, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$; query $\Pi.\mathsf{asySHK}_0$ with $(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$, obtain K_0;
   return K_0;
Case 3: if $\exists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0}$ or $\exists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}}$,
  Subcase 3.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,

query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 3.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,
query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
Subcase 3.3 : otherwise, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_1}$; query $\Pi.\mathsf{asySHK}_1$ with $(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$, obtain K_1;
return K_1;
Case 4: if $\nexists(\mathrm{SK}_1, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0} \cup T_{P_1}$ or $\nexists(*, *, \mathrm{pk}_1, \mathrm{PK}_1) \in T_{P_0^{-1}} \cup T_{P_1^{-1}}$,
Subcase 4.1 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_0}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_0;
return K_0;
Subcase 4.2 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, *, \mathrm{PK}_0) \in T_{H_1}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K_1;
return K_1;
Subcase 4.3 : if $\exists(\mathrm{SK}_0, \mathrm{sk}_0, \mathrm{pk}_0, \mathrm{PK}_0) \in T_{\mathrm{symKG}}$,
run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K;
return K;
//or run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$, query $\Pi.\mathsf{asySHK}_1$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$, obtain K; return K;
Subcase 4.4 : otherwise, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$, record $(\tilde{\mathrm{SK}}_0, \mathrm{sk}_0, *, \Pi.\mathsf{asyKG}_0(\tilde{\mathrm{SK}}_0))$ in T_{H_0}; run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_1}(\mathrm{pk}_1)$; query $\Pi.\mathsf{asySHK}_0$ with $(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$, obtain K_0;
return K_0. *//or, $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_1}$; record $(\tilde{\mathrm{SK}}_0, \mathrm{sk}_0, *, \Pi.\mathsf{asyKG}_1(\tilde{\mathrm{SK}}_0))$ in T_{H_1};*

// run $\mathrm{PK}_1 \leftarrow \mathcal{S}^{P_0}(\mathrm{pk}_1)$; then query $\Pi.\mathsf{asySHK}_1$ with $(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$, obtain K_1; return K_1;

Claim 10. Game 8 $\approx$ Game 9.

Proof.

The differences between Game 8 and Game 9 occurs in the Subcase 2.3, the Subcase 3.3 and the Subcase 4.4.

For the Subcase 2.3, in Game 8, the simulator responds to a query $(\mathrm{sk}_0, \mathrm{pk}_1)$ with $\mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$, while in Game 9, the simulator responds $K_0 \leftarrow \Pi.\mathsf{asySHK}_0(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$ for a random sampled $\tilde{\mathrm{SK}}_0$. The distributions of $\mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$ and $\Pi.\mathsf{asySHK}_0(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$ are indistinguishable, both satisfying the consistency conditions.

For the Subcase 3.3, the analysis is similar to that of the Subcase 2.3.

For the Case 4, in Game 8, the simulator responds to a query $(\mathrm{sk}_0, \mathrm{pk}_1)$ with $\mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$, while in Game 9, the simulator responds with $K_0 \leftarrow \Pi.\mathsf{asySHK}_0(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$ for a random sampled $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_0}$ and $\mathrm{PK}_1 = \mathcal{S}^{P_1}(\mathrm{pk}_1)$ (or responds with $K_1 \leftarrow \Pi.\mathsf{asySHK}_1(\tilde{\mathrm{SK}}_0, \mathrm{PK}_1)$ for a random sampled $\tilde{\mathrm{SK}}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{SK}_1}$ and $\mathrm{PK}_1 = \mathcal{S}^{P_0}(\mathrm{pk}_1)$). The distributions of $\mathsf{I.symSHK}(\mathrm{sk}_0, \mathrm{pk}_1)$ and K_0 (or K_1) are indistinguishable, both satisfying the consistency conditions.

By definition, the responses of $P_0, P_1, P_0^{-1}, P_1^{-1}, H_0, H_1, \mathsf{symKG}$ queries are identical in Game 8 and Game 9.

Let Bad_9 denote the event that before the symSHK query $(\mathrm{sk}_0, \mathrm{pk}_1)$, the adversary makes a query $\tilde{\mathrm{SK}}_0$ on H_0 or makes a query $P_1^{-1}(\mathrm{pk}_1)$ on P_1 such that $\mathcal{S}^{H_0}(\tilde{\mathrm{SK}}_0) = \mathrm{sk}_0' \neq \mathrm{sk}_0$ or $\mathcal{S}^{P_1}(P_1^{-1}(\mathrm{pk}_1)) = \mathrm{pk}_1' \neq \mathrm{pk}_1$. The responses of symSHK queries are consistent if Bad_9 did not occur. Since $\tilde{\mathrm{SK}}_0$ and $P_1^{-1}(\mathrm{pk}_1)$ are random strings unknown to the adversary, the only way the adversary can get the two strings is random guessing, and the probability of guessing right (Bad_9 occurs) is bound by $\frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_1}|}$.

Combing together, we have

$$|\Pr[\text{Game } 8 = 1] - \Pr[\text{Game } 9 = 1]| \leq q \cdot \left(\frac{q}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q}{|\mathsf{DomAsy}_{\mathrm{PK}_1}|}\right) \leq \mathsf{negl}(\lambda)$$

Game 10. In Game 9, the queries to the adversarial interfaces are answered by the tables which are maintained by the simulator and by making queries to $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1$. The simulator never make queries directly to $H_0, H_1, P_0, P_1, P_0^{-1}, P_1^{-1}, \mathsf{I.symKG}, \mathsf{I.symSHK}$; these oracles are *only* used to answer the $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1$ queries (either generated by the adversary or by the simulator). At this point, we can replace the calls to $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1$ with the calls to ideal algorithms $\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0$, $\mathsf{I.asySHK}_1$ respectively, resulting in Game 10.

We note that in Game 9, the simulator is efficient, and it responds to the adversarial interfaces just by keeping several tables and calling $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1$ at the honest interfaces. Thus, we can build a simulator that responds to the honest and adversarial queries precisely as the simulator does in Game 9, except for the changes in calls to the honest interfaces. Hence, the adversary's view in Game 10 is identical to the ideal world and it suffices to prove that any adjacent games are indistinguishable. Next we give the rigorous proof for the indistinguishably between each adjacent games.

Claim 11. Game 9 $\approx$ Game 10.

Proof. Let $(\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$ be the function pair that samples from $\mathcal{T}_{\mathsf{asyNIKE}}$.We note that in Game 9, the simulator responds all of the adversarial interfaces just using the tables and the algorithms $(\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1)$ at honest interfaces, and it never directly calls the real oracles at honest interfaces. We immediately observe that the simulator in Game 10 is identical to the simulator in the ideal game, which refers to

$$\Pr[\text{Game } 10 = 1] = \Pr[\text{Ideal Game} = 1]$$

Therefore, it is rest to prove that Game 9 and Game 10 are close. H_0, H_1 are random oracles, $P_0, P_0^{-1}, P_1, P_1^{-1}$ are random permutations, $(\mathsf{I.symKG}, \mathsf{I.symSHK})$ is an ideal symmetric NIKE with $\mathsf{I.symKG}, \mathsf{I.symSHK}$ being random injections except with shared key property $\mathsf{symSHK}(sk_0, \mathsf{symKG}(sk_1)) = \mathsf{symSHK}(\mathsf{symKG}(sk_0), sk_1)$.

Conditioned on the oracles H_0, H_1 having no collisions, with regard to the asymmetric key generation oracles, for any SK_0 and SK_1, it's oblivious that the distributions of $\Pi.\mathsf{asyKG}_0(\mathrm{SK}_0)$ and $\mathsf{I.asyKG}_0(\mathrm{SK}_0)$ are identical, and the distributions of $\Pi.\mathsf{asyKG}_1(\mathrm{SK}_1)$ and $\mathsf{I.asyKG}_1(\mathrm{SK}_1)$ are identical; and with regard to the asymmetric shared key oracles, for any $(\mathrm{SK}_0, \mathrm{PK}_1)$, the distributions of $\Pi.\mathsf{asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$ and $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$ are identical; in addition, for any $(\mathrm{SK}_1, \mathrm{PK}_0)$ the distributions of $\Pi.\mathsf{asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0)$ and $\mathsf{I.asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0)$ are identical.

That the oracle H_0 has collision means there are two queries SK_0 and SK_0' to H_0 such that $\mathrm{SK}_0 \neq \mathrm{SK}_0'$ and $H_0(\mathrm{SK}_0) = H_0(\mathrm{SK}_0')$. That the oracle H_1 has collision means there are two queries SK_1 and SK_1' to H_1 such that $\mathrm{SK}_1 \neq \mathrm{SK}_1'$ and $H_1(\mathrm{SK}_1) = H_1(\mathrm{SK}_1')$;

If none of the collision occurs, we can replace $(\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1)$ with $(\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$, which represents Game 10. Moreover, we can bound the probability of H_0, H_1 having collision by

$$\Pr[\mathsf{Collision}] \leq \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_1}|} \leq \mathsf{negl}(\lambda),$$

which refers to

$$|\Pr[\text{Game } 9 = 1] - \Pr[\text{Game } 10 = 1]| \leq \mathsf{negl}(\lambda)$$

Simulator In Ideal Game. Let $(\mathsf{I}.\mathrm{asyKG}_0, \mathsf{I}.\mathrm{asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$ be the function pair that samples from the ideal asymmetric NIKE family $\mathcal{T}_{\mathsf{asyNIKE}}$, the simulator works as follows. In Game 10, the simulator in the ideal game maintains eight tables in the same way as in Game 9 except that the table entries set as the responses of $\Pi.\mathsf{asyKG}_0, \Pi.\mathsf{asyKG}_1, \Pi.\mathsf{asySHK}_0, \Pi.\mathsf{asySHK}_1$ are replaced by the responses of $\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$, respectively.

By definition, the simulator $\mathcal{S}$ now has access to $(\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.SHK}_0, \mathsf{I}.\mathrm{SHK}_1)$ at the honest interfaces.

And for the adversarial queries, $\mathcal{S}$ works the same as in Game 9, by just using the tables and querying the honest interfaces.

Combining all claims together, we have

$$\begin{aligned} |\Pr[\text{Real Game} = 1] - \Pr[\text{Ideal Game} = 1]| &\leq \frac{2q^2}{|\mathsf{DomSym}_{\mathrm{sk}}|} + \frac{4q^2}{|\mathsf{DomSym}_{\mathrm{pk}}|} \\ &+ \frac{3q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{2q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_1}|} + \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} + \frac{2q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_1}|} \\ &\leq \mathsf{negl}(\lambda), \end{aligned}$$

thus we complete the entire proof.

B Proof of Theorem 2

Proof. Here, we give full proof of Theorem 2, that the constructed $\Pi.\mathsf{EOT}$ in Sec.?? is indifferentiable from ideal $\mathsf{I.EOT}$.

In the real world, the differentiator $\mathcal{D}$ has oracle access to $(\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$ via the honest interface and oracle access to $(H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$ via the adversarial interface. In contrast, in the ideal world, the differentiator $\mathcal{D}$ has oracle access to $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface. Therefore, to establish a proof, we need to build an explicit (and efficient) simulator $\mathcal{S}$ that simulates the rest oracles $(H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1)$ properly by making queries to $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$.

Namely, for any PPT differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real game is computationally close to the view in the ideal game. To do so, we will go through with a sequence of hybrid games, where in each game, the simulator responds to all of the queries (both honest and adversarial) in a slightly different way and the last game is the same as the ideal world. Note that the differentiator $\mathcal{D}$ can make at most q queries to the oracles, where $q = \text{poly}(\lambda)$.

First, we describe our simulator $\mathcal{S}$ in the ideal game.

Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ has the external oracle access to the ideal random OT scheme $\mathsf{I.EOT} = (\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_2})$. The simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\mathcal{S}^{H_0}(\text{SK}_0, b)$:

Case 1: if $\exists(\text{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
return Q_{1-b};
Case 2: otherwise, query the external $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK_0, b), and obtain Q_0, Q_1;
$T_{H_0} = T_{H_0} \cup (\text{SK}_0, b, Q_{1-b}, Q_b)$;
return Q_{1-b}.

$\mathcal{S}^{H_1}(Q_d)$:

Case 1: if $\exists(Q_d, \tilde{k}) \in T_{H_1}$,
return $\tilde{k}$;
Case 2: if $\exists(\text{SK}_0, \text{PK}_0) \in T_{\text{asyKG}_0} \wedge (\text{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_b, \tilde{k}, \text{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$, s.t. $Q_d = Q_{1-b}$,
return $\tilde{k}$;
Case 3: otherwise, $\tilde{k} \leftarrow \{0,1\}^{\ell_2(\lambda)}$; $T_{H_1} = T_{H_1} \cup (Q_d, \tilde{k})$;
return $\tilde{k}$.

$\mathcal{S}^{\mathcal{E}}(\text{PK}_0, \tilde{k})$:

Case 1: if $\exists(Q_b, \tilde{k}, \text{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return Q_b;

Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$ and $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
return Q_b;
Subcase 2.2: else, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 0)$, obtain (Q_0, Q_1);
if $Q_1 = Q_{1-b}$, return Q_0; else, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 1)$, obtain (Q'_0, Q'_1),
if $Q'_0 = Q_{1-b}$, return Q'_1; else, $\tilde{Q}_b \leftarrow \{0,1\}^{\ell_2(\lambda)}$, $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (\tilde{Q}_b, \tilde{k}, \mathrm{PK}_0)$;
return $\tilde{Q}_b$.
Case 3 : otherwise, randomly sample $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$, $b \leftarrow \{0,1\}$, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK_0, b), obtain (Q_0, Q_1); implicitly set $\mathcal{S}^{H_1}(Q_{1-b}) = \tilde{k}$ and $\mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0) = \mathrm{K}_0$; $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (Q_b, \tilde{k}, \mathrm{PK}_0)$;
return Q_b.

$\mathcal{S}^{\mathcal{E}^{-1}}(Q_b, \tilde{k})$:

Case 1: if $\exists(Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return PK_0;
Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$ and $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$, return PK_0;
Subcase 2.2: else, run $\mathrm{PK}_0 = \mathcal{S}^{\mathsf{asyKG}_0}(\mathrm{SK}_0)$, return PK_0;
Case 3 : $\mathrm{PK}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{PK}_0}$, $T_{\mathcal{E}^{-1}} = T_{\mathcal{E}^{-1}} \cup (Q_b, \tilde{k}, \mathrm{PK}_0)$;
return PK_0.

$\mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0)$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
return PK_0;
Case 2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return PK_0;
Case 3: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$ and $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ s.t. $\mathrm{PK}_1 = \mathsf{I.EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$,
return PK_0;
Case 4: otherwise, $\mathrm{PK}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{PK}_0}$; $T_{\mathrm{asyKG}_0} = T_{\mathrm{asyKG}_0} \cup (\mathrm{SK}_0, \mathrm{PK}_0)$;
return PK_0.

$\mathcal{S}^{\mathrm{asyKG}_1}(\mathrm{SK}_1)$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,
return PK_1;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0} \wedge \exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ and $\mathrm{PK}_0 = \mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0)$,
return PK_1;
Case 3: otherwise, query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1;
return PK_1.

$\mathcal{S}^{\mathrm{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1)$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,

query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Case 4: otherwise, $b \leftarrow \{0,1\}$, query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
$T_{\mathrm{asySHK}_0} = T_{\mathrm{asySHK}_0} \cup (\mathrm{SK}_0, \mathrm{PK}_1, K_b)$;
return K_b.

$\mathcal{S}^{\mathrm{asySHK}_1}(\mathrm{SK}_1, \mathrm{PK}_0)$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Subcase 2.2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Subcase 2.3: otherwise, query $\Pi.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1; $b \leftarrow \{0,1\}$,
query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
return K_b;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Case 4: otherwise, randomly sample $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}, b \leftarrow \{0,1\}$, query $\Pi.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1;
query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
return K_b.

Next, we describe our simulator $\mathcal{S}$ in the hybrid games.

Game 0. This game is identical to the real game except that the simulator maintains eight tables for the adversarial interfaces, referring to H_0 table T_{H_0}, H_1 table T_{H_1}, $\mathsf{I.asyKG}_0$ table T_{asyKG_0}, $\mathsf{I.asyKG}_1$ table T_{asyKG_1}, $\mathsf{I.asySHK}_0$ table T_{asySHK_0} and $\mathsf{I.asySHK}_1$ table T_{asySHK_1} , $\mathcal{E}$ table $T_{\mathcal{E}}$ and $\mathcal{E}^{-1}$ table $T_{\mathcal{E}^{-1}}$. The tables are initially empty, and the table entries are added by the simulator when new queries are answered, in the following forms:

- $T_{H_0} := (\mathrm{SK}_0, b, Q_b, Q_{1-b})$ with $b \in \{0,1\}$;
- $T_{H_1} := (Q_d, \tilde{k})$;
- $T_{\mathcal{E}} := (Q_b, \tilde{k}, \mathrm{PK}_0)$;
- $T_{\mathcal{E}^{-1}} := (Q_b, \tilde{k}, \mathrm{PK}_0)$;
- $T_{\mathrm{asyKG}_0} := (\mathrm{SK}_0, \mathrm{PK}_0)$;
- $T_{\mathrm{asyKG}_1} := (\mathrm{SK}_1, \mathrm{PK}_1)$;
- $T_{\mathrm{asySHK}_0} := (\mathrm{SK}_0, \mathrm{PK}_1, K_0)$;
- $T_{\mathrm{asySHK}_1} := (\mathrm{SK}_1, \mathrm{PK}_0, K_1)$.

Concretely, the simulator responds to the queries by forwarding the responses of the corresponding oracles, such that the simulator's responses are the same as in the real world. For instance, $\mathcal{S}_0^{H_0}(\mathrm{SK}_0, b) = H_0(\mathrm{SK}_0, b)$, $\mathcal{S}_0^{H_1}(Q_d) = H_1(Q_d)$, $\mathcal{S}_0^{\mathrm{asyKG}_0}(\mathrm{SK}_0) = \mathsf{I.asyKG}_0(\mathrm{SK}_0)$, $\mathcal{S}_0^{\mathrm{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1) = \mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$ and so forth.

In Game 0, the simulator maintains the tables as follows.

1. H_0-table T_{H_0}: Once the adversary queries oracle H_0 with (SK_0, b) which does not exist in T_{H_0}, the simulator inserts $(\mathrm{SK}_0, b, H_0(\mathrm{SK}_0, b), *)$ into the T_{H_0} table.
2. H_1-table T_{H_1}: Once the adversary queries oracle H_1 with Q_d which does not exist in T_{H_1}, the simulator inserts $(Q_d, H_1(Q_d))$ into the T_{H_1}-table.
3. $\mathsf{I.asyKG}_0$-table T_{asyKG_0}: Once the adversary queries $\mathsf{I.asyKG}_0$ with SK_0 which does not exist in T_{asyKG_0}-table, the simulator inserts $(\mathrm{SK}_0, \mathsf{I.asyKG}_0(\mathrm{SK}_0)$ into the T_{asyKG_0}-table.
4. $\mathsf{I.asyKG}_1$-table T_{asyKG_1}: Once the adversary queries $\mathsf{I.asyKG}_1$ with SK_1 which does not exist in T_{asyKG_1}-table, the simulator inserts $(\mathrm{SK}_1, \mathsf{I.asyKG}_1(\mathrm{SK}_1)$ into the T_{asyKG_1}-table.
5. $\mathsf{I.asySHK}_0$-table T_{asySHK_0}: Once the adversary queries $\mathsf{I.asySHK}_0$ with $(\mathrm{SK}_0, \mathrm{PK}_1)$ which does not exist in the T_{asySHK_0}-table, the simulator inserts $(\mathrm{SK}_0, \mathrm{PK}_1, \mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1))$ into T_{asySHK_0}-table.
6. $\mathsf{I.asySHK}_1$-table T_{asySHK_1}: Once the adversary queries $\mathsf{I.asySHK}_1$ with $(\mathrm{SK}_1, \mathrm{PK}_0)$ which does not exist in the T_{asySHK_1}-table, the simulator inserts $(\mathrm{SK}_1, \mathrm{PK}_0, \mathsf{I.asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0))$ into T_{asySHK_1}-table.

At this point all the queries are responded by the real oracles and these tables are just keeping track of information related to $\mathcal{D}$'s queries (to the adversarial interfaces) and completely hidden to the adversary, hence the adversary's view in real game is identical to the one in Game 0.

Next, we illustrate an alternative way to answer part of the queries, by using these tables and the honest interfaces.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries at adversarial interfaces. Specifically, the simulator $\mathcal{S}_1$ has the external oracle access to the random OT scheme $\Pi.\mathsf{EOT} = (\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$. The simulator $\mathcal{S}_1$ will provide the following interfaces for the external differentiator $\mathcal{D}$, as shown in the box of Simulator $\mathcal{S}_1$.

Compared to Game 0, in Game 1 the simulator keeps a longer table, and for part of the queries, the simulator responds to them in an alternative way, which is only using the tables and the honest interfaces. Moreover, in Game 1, the tuples stored in the tables are consistent with the responses by the real oracles to the adversary's queries. Hence, in either game, the response of any query is identical, which refers to that the view in Game 1 is identical to the one in Game 0.

However, the simulator can only answer part of the queries by tables and honest interfaces, and for the rest it has to call the real oracles. Thus, in the following hybrid games, we will illustrate additional alternative ways (not calling the real oracles) to respond to the rest queries, without changing the adversary's view significantly.

Simulator $\mathcal{S}_1$

$\underline{\mathcal{S}_1^{H_0}(\mathrm{SK}_0, b)}$:

Case 1: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, *) \in T_{H_0}$,
return Q_{1-b};
Case 2: otherwise, $Q_{1-b} = H_0(\mathrm{SK}_0, b)$; $T_{H_0} = T_{H_0} \cup (\mathrm{SK}_0, b, Q_{1-b}, Q_b)$;
return Q_{1-b}.

$\underline{\mathcal{S}_1^{H_1}(Q_d)}$:

$\underline{\mathcal{S}^{H_1}(Q_d)}$:

Case 1: if $\exists(Q_d, \tilde{k}) \in T_{H_1}$,
return $\tilde{k}$;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0} \wedge (\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$, s.t. $Q_d = Q_{1-b}$,
return $\tilde{k}$;
Case 3: otherwise, $\tilde{k} = H_1(Q_d)$; $T_{H_1} = T_{H_1} \cup (Q_d, \tilde{k})$;
return $\tilde{k}$.

$\underline{\mathcal{S}_1^{\mathcal{E}}(\mathrm{PK}_0, \tilde{k})}$:

Case 1: if $\exists(Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return Q_b;
Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$ and $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
return Q_b;
Subcase 2.2: else, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 0)$, obtain (Q_0, Q_1);
if $Q_1 = Q_{1-b}$, return Q_0; else, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 1)$, obtain (Q_0', Q_1'),
if $Q_0' = Q_{1-b}$, return Q_1'; else, $\tilde{Q}_b = \mathcal{E}(\mathrm{PK}_0, \tilde{k})$; $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (\tilde{Q}_b, \tilde{k}, \mathrm{PK}_0)$;
return $\tilde{Q}_b$.
Case 3 : otherwise, $\tilde{Q}_b = \mathcal{E}(\mathrm{PK}_0, \tilde{k})$;$T_{\mathcal{E}} = T_{\mathcal{E}} \cup (\tilde{Q}_b, \tilde{k}, \mathrm{PK}_0)$;
return $\tilde{Q}_b$.

$\underline{\mathcal{S}_1^{\mathcal{E}^{-1}}(Q_b, \tilde{k})}$:

Case 1: if $\exists(Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return PK_0;
Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$ and $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$ return PK_0;
Subcase 2.2: else, run $\mathrm{PK}_0 = \mathcal{S}_1^{\mathsf{asyKG}_0}(\mathrm{SK}_0)$, return PK_0;
Case 3 : $\mathrm{PK}_0 = \mathcal{E}^{-1}(Q_b, \tilde{k})$; $T_{\mathcal{E}^{-1}} = T_{\mathcal{E}^{-1}} \cup (Q_b, \tilde{k}, \mathrm{PK}_0)$;
return PK_0.

$\underline{\mathcal{S}_1^{\mathrm{asyKG}_0}(\mathrm{SK}_0)}$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
return PK_0;
Case 2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return PK_0;
Case 3: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$ and $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ s.t. $\mathrm{PK}_1 = \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$,

return PK_0;
Case 4: otherwise, $\mathrm{PK}_0 = \mathsf{I.asyKG_0}(\mathrm{SK}_0)$; $T_{\mathrm{asyKG}_0} = T_{\mathrm{asyKG}_0} \cup (\mathrm{SK}_0, \mathrm{PK}_0)$;
return PK_0.

$\underline{\mathcal{S}_1^{\mathrm{asyKG}_1}(\mathrm{SK}_1)}$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,
return PK_1;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0} \wedge \exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ and $\mathrm{PK}_0 = \mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0)$,
return PK_1;
Case 3: otherwise, $\mathrm{PK}_1 = \mathsf{I.asyKG_1}(\mathrm{SK}_1)$; $T_{\mathrm{asyKG}_1} = T_{\mathrm{asyKG}_1} \cup (\mathrm{SK}_1, \mathrm{PK}_1)$;
return $mathrmPK_1$.

$\underline{\mathcal{S}_1^{\mathrm{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1)}$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,
query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Case 4: otherwise, $K_0 = \mathsf{I.asySHK_0}(\mathrm{SK}_0, \mathrm{PK}_1)$; $T_{\mathrm{asySHK}_0} = T_{\mathrm{asySHK}_0} \cup (\mathrm{SK}_0, \mathrm{PK}_1, K_0)$;
return K_0.

$\underline{\mathcal{S}_1^{\mathrm{asySHK}_1}(\mathrm{SK}_1, \mathrm{PK}_0)}$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Subcase 2.2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Subcase 2.3: else, query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1; $b \leftarrow \{0, 1\}$, query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
return K_b;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
query $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Case 4: otherwise, $K_b = \mathsf{I.asySHK_1}(\mathrm{SK}_1, \mathrm{PK}_0)$;
return K_b.

Game 2. This game is identical to Game 1, except for responding to H_0 queries. The simulator responds to a query SK_0 on H_0 as follows:

Simulator $\mathcal{S}_2$

$\underline{\mathcal{S}_2^{H_0}(\mathrm{SK}_0)}$:

Case 1: if $\exists(\mathrm{SK}_0, *, Q_{1-b}, *) \in T_{H_0}$,
 return Q_{1-b};
Case 2: otherwise, query the external $\Pi.\mathsf{EOT}_{\mathsf{A}_1}$ with (SK_0, b), and obtain Q_0, Q_1;
$T_{H_0} = T_{H_0} \cup (\mathrm{SK}_0, b, Q_{1-b}, Q_b)$;
 return Q_{1-b}.

The only difference between Game 1 and Game 2 occurs in the Case 2 where (SK_0, b) never appears in T_{H_0}. In Game 1, the simulator responds with $H_0(\mathrm{SK}_0, b)$ while in Game 2, the simulator responds with $Q_{1-b} = \mathsf{LoR}_{1-b}(\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b))$. By definition, $H_0(\mathrm{SK}_0)$ is identical to Q_{1-b}, from the adversary $\mathcal{D}$'s view, Game 1 and Game 2 are identical.

Game 3. This game is identical to Game 2, except for responding to H_1 queries. The simulator responds to a query Q_d on H_1 as in the box of Simulator $\mathcal{S}_3$.

The only difference between Game 2 and Game 3 occurs in the Case 3 of a H_0 query Q_d which never appears in $T_{H_1} \cup T_{H_0}$.

In Game 2, the simulator responds to H_1 query on Q_d with $H_1(Q_d)$; while in Game 3, the simulator responds with a random string $\tilde{k}$. Due to definition, the only case that the simulator queries H_1 with such Q_d is when the adversary $\mathcal{D}$ knows nothing of $H_1(Q_d)$, although the adversary might know $\Pi.\mathrm{OT}_{\mathrm{B}_2}(\mathrm{SK}_1, Q_d, Q_{1-d})$. Therefore, from the adversary $\mathcal{D}$'s view, $H_1(Q_d)$ is uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, which implies $\mathcal{D}$'s view in Game 3 is indistinguishable from its view in Game 2 with high probability.

Simulator $\mathcal{S}_3$

$\underline{\mathcal{S}_3^{H_1}(Q_d)}$:

Case 1: if $\exists(Q_d, \tilde{k}) \in T_{H_1}$,
 return $\tilde{k}$;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0} \wedge (\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$, s.t. $Q_d = Q_{1-b}$,
 return $\tilde{k}$;
Case 3: otherwise, $\tilde{k} \leftarrow \{0,1\}^{\ell_2(\lambda)}$; $T_{H_1} = T_{H_1} \cup (Q_d, \tilde{k})$;
 return $\tilde{k}$.

Game 4. This game is identical to Game 3, except for responding to $\mathsf{I.asyKG}_0$ queries. The simulator responds to a query SK_0 at $\mathsf{I.asyKG}_0$ as in the box of $\mathcal{S}_4$.

The only difference between Game 3 and Game 4 occurs in the Case 4 of a $\mathsf{I.asyKG}_0$ query. In Game 3, the simulator responds to a $\mathsf{I.asyKG}_0$ query SK_0

with $\mathsf{I.asyKG_0}(\mathrm{SK}_0)$; while in Game 4, the simulator responds with a randomly sampled string $\mathrm{PK}_0 \leftarrow \{0,1\}^{\ell_2(\lambda)}$.

Due to definition, the only case that the simulator queries $\mathsf{I.asyKG_0}$ with SK_0 is when the adversary $\mathcal{D}$ knows nothing of $\mathsf{I.asyKG_0}(\mathrm{SK}_0)$, although the adversary might know $\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)$ for $b = 0, 1$. Therefore, from the adversary $\mathcal{D}$'s view, $\mathsf{I.asyKG_0}(\mathrm{SK}_0)$ is uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$. Note that $\mathcal{S}_1^{H_1}(Q_1)$ is also uniformly distributed in $\{0,1\}^{\ell_2(\lambda)}$, which implies $\mathcal{D}$'s view in Game 4 and $\mathcal{D}$'s view in Game 3 are indistinguishable.

Simulator $\mathcal{S}_4$

$\mathcal{S}_4^{\mathrm{asyKG}_0}(\mathrm{SK}_0)$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
 return PK_0;
Case 2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
 return PK_0;
Case 3: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$ and $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ s.t. $\mathrm{PK}_1 = \Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$,
 return PK_0;
Case 4: otherwise, $\mathrm{PK}_0 \leftarrow \{0,1\}^{\ell_2(\lambda)}$; $T_{\mathrm{asyKG}_0} = T_{\mathrm{asyKG}_0} \cup (\mathrm{SK}_0, \mathrm{PK}_0)$;
 return PK_0.

Game 5. This game is identical to Game 4, except for responding to $\mathsf{I.asyKG_1}$ queries. The simulator responds to a query SK_1 on $\mathsf{I.asyKG_1}$ as in the box of Simulator $\mathcal{S}_5$.

The only difference between Game 4 and Game 5 occurs in the Case 3 of a $\mathsf{I.asyKG_1}$ query. In Game 4, the simulator responds to a $\mathsf{I.asyKG_1}$ query SK_1 with $\mathsf{I.asyKG_1}(\mathrm{SK}_1)$; while in Game 5, the simulator responds with $\Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$. Due to definition, $\mathsf{I.asyKG_1}(\mathrm{SK}_1)$ and $\Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$ are identical for any SK_1. Therefore, $\mathcal{D}$'s view in Game 5 and $\mathcal{D}$'s view in Game 4 are the same.

Simulator $\mathcal{S}_5$

$\mathcal{S}_5^{\mathrm{asyKG}_1}(\mathrm{SK}_1)$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,
 return PK_1;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0} \wedge \exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$ and $\mathrm{PK}_0 = \mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0)$,
 return PK_1;
Case 3: otherwise, query $\Pi.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1;
 return PK_1.

Game 6. This game is identical to Game 5, except for responding to $\mathcal{E}$ queries. The simulator responds to a query $(\mathrm{PK}_0, \tilde{k})$ on $\mathcal{E}$ as follows:

Simulator $\mathcal{S}_6$

$\underline{\mathcal{S}_6^{\mathcal{E}}(\mathrm{PK}_0, \tilde{k})}$:

Case 1: if $\exists(Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return Q_b;
Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$ and $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
return Q_b;
Subcase 2.2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$ and $\nexists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 0)$, obtain (Q_0, Q_1),
if $Q_1 = Q_{1-b}$, return Q_0;
else, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $(\mathrm{SK}_0, 1)$, obtain (Q_0', Q_1');
and if $Q_0' = Q_{1-b}$, return Q_1';
else, $\tilde{Q_b} \leftarrow \{0,1\}^{\ell_2(\lambda)}$, $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (\tilde{Q_b}, \tilde{k}, \mathrm{PK}_0)$; return $\tilde{Q_b}$.
Case 3 : otherwise, randomly sample $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$, $b \leftarrow \{0,1\}$, query $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK_0, b), obtain (Q_0, Q_1); implicitly set $\mathcal{S}^{H_1}(Q_{1-b}) = \tilde{k}$ and $\mathcal{S}^{\mathrm{asyKG}_0}(\mathrm{SK}_0) = \mathrm{PK}_0$; $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (Q_b, \tilde{k}, \mathrm{PK}_0)$;
return Q_b.

The only difference between Game 5 and Game 6 occurs in the Case 3 of a $\mathcal{E}$ query. In Game 5, the simulator responds to a $\mathcal{E}$ query $(\mathrm{PK}_0, \tilde{k})$ with $\mathcal{E}(\mathrm{PK}_0, \tilde{k})$; while in Game 6, the simulator responds with $\mathsf{LoR}_b(\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b))$ for a random bit b.

Game 7. This game is identical to Game 6, except for responding to $\mathcal{E}^{-1}$ queries. The simulator responds to a query $(Q_b, \tilde{k})$ on $\mathcal{E}^{-1}$ as follows:

Simulator $\mathcal{S}_7$

$\underline{\mathcal{S}_7^{\mathcal{E}^{-1}}(Q_b, \tilde{k})}$:

Case 1: if $\exists(Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return PK_0;
Case 2: if $\exists(Q_{1-b}, \tilde{k}) \in T_{H_1}$ and $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$, return PK_0;
Subcase 2.2: run $\mathrm{PK}_0 = \mathcal{S}^{\mathsf{asyKG}_0}(\mathrm{SK}_0)$, return PK_0;
Case 3 : $\mathrm{PK}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{PK}_0}$, $T_{\mathcal{E}^{-1}} = T_{\mathcal{E}^{-1}} \cup (Q_b, \tilde{k}, \mathrm{PK}_0)$;
return PK_0.

The only difference between Game 6 and Game 7 occurs in the Case 3 of a $\mathcal{E}^{-1}$ query. In Game 6, the simulator responds to a $\mathcal{E}^{-1}$ query $(Q_b, \tilde{k})$ with

$\mathcal{E}^{-1}(Q_b, \tilde{k})$; while in Game 7, the simulator responds with a randomly sampled string $\mathrm{PK}_0 \leftarrow \mathsf{DomAsy}_{\mathrm{PK}_0}$.

Game 8. This game is identical to Game 7, except for responding to $\mathsf{I.asySHK}_0$ queries. The simulator responds to a query $(\mathrm{SK}_0, \mathrm{PK}_1)$ on $\mathsf{I.asySHK}_0$ as follows.

Simulator $\mathcal{S}_8$

$\mathcal{S}_8^{\mathrm{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1)$:

Case 1: if $\exists(\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1}$,
query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Case 4: otherwise, $b \leftarrow \{0,1\}$, query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
return K_b.

The only difference between Game 7 and Game 8 occurs in the Case 4.

In Game 7, the simulator responds to a $\mathsf{I.asySHK}_0$ query $(\mathrm{SK}_0, \mathrm{PK}_1)$ with $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$; while in Game 8, the simulator responds with $\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ for a random bit b.

By definition, $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$ and $\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ are identical for any $(\mathrm{SK}_0, \mathrm{PK}_1)$. Therefore, $\mathcal{D}$'s view in Game 8 and $\mathcal{D}$'s view in Game 7 are identical.

Simulator $\mathcal{S}_9$

$\mathcal{S}_9^{\mathrm{asySHK}_1}(\mathrm{SK}_1, \mathrm{PK}_0)$:

Case 1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_0, K) \in T_{\mathrm{asySHK}_1}$,
return K;
Case 2: if $\exists(\mathrm{SK}_0, \mathrm{PK}_0) \in T_{\mathrm{asyKG}_0}$,
Subcase 2.1: if $\exists(\mathrm{SK}_1, \mathrm{PK}_1) \in T_{\mathrm{asyKG}_1} \wedge (\mathrm{SK}_0, \mathrm{PK}_1, K) \in T_{\mathrm{asySHK}_0}$,
return K;
Subcase 2.2: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0}$,
query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;
Subcase 2.3: otherwise, query $\Pi.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1; $b \leftarrow \{0,1\}$,
query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b;
return K_b;
Case 3: if $\exists(\mathrm{SK}_0, b, Q_{1-b}, Q_b) \in T_{H_0} \wedge (Q_{1-b}, \tilde{k}) \in T_{H_1} \wedge (Q_b, \tilde{k}, \mathrm{PK}_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
query $\Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with $(Q_0, Q_1, \mathrm{SK}_1)$, obtain K_0, K_1;
return K_b;

Case 4: otherwise, randomly sample $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}, b \leftarrow \{0,1\}$, query $\Pi.\mathsf{EOT}_{\mathsf{B}_1}$ with SK_1, obtain PK_1; query $\Pi.\mathsf{EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, \mathrm{PK}_1, b)$, obtain K_b; return K_b.

Game 9. This game is identical to Game 8, except for responding to $\mathsf{I.asySHK}_1$ queries. The simulator responds to a query $(\mathrm{SK}_1, \mathrm{PK}_0)$ on $\mathsf{I.asySHK}_1$ as follows.

The only difference between Game 8 and Game 9 occurs in the Case 4. In Game 8, the simulator responds to a random query $(\mathrm{SK}_1, \mathrm{PK}_0)$ to $\mathsf{I.asySHK}_1$ with $\mathsf{I.asySHK}_1(\mathrm{SK}_1, \mathrm{PK}_0)$; while in Game 9, the simulator responds with $\mathsf{LoR}_b(\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b))$ for a randomly sampled string $\mathrm{SK}_0 \leftarrow \{0,1\}^{\ell_1(\lambda)}$ and a random bit $b \leftarrow \{0,1\}$.

Due to definition, the only case that the simulator queries $\mathsf{I.asySHK}_1$ with $(\mathrm{SK}_1, \mathrm{PK}_0)$ is when the adversary $\mathcal{D}$ knows nothing of $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$, although the adversary might know $\Pi.\mathrm{OT}_{\mathrm{B}_1}(\mathrm{SK}_1)$. Therefore, from the adversary $\mathcal{D}$'s view, $\mathsf{I.asySHK}_0(\mathrm{SK}_0, \mathrm{PK}_1)$ is uniformly distributed in $\{0,1\}^{n_3(\lambda)}$. Note that K is also uniformly distributed in $\{0,1\}^{n_3(\lambda)}$, which implies $\mathcal{D}$'s view in Game 9 and $\mathcal{D}$'s view in Game 8 are indistinguishable.

Game 10. In Game 9, the queries to the adversarial interfaces are answered by the tables which are maintained by the simulator and by making queries to $\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2}$. The simulator never make queries directly to $H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$; these oracles are *only* used to answer the $\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2}$ queries (either generated by the adversary or by the simulator's response to $H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ queries). At this point, we can replace the calls to $\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2}$ with the calls to ideal algorithms $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}$, $\mathsf{I.EOT}_{\mathsf{B}_2}$ respectively, resulting in Game 10.

Note that in Game 9, the simulator is efficient, and it responds to the adversarial interfaces just by keeping several tables and calling $\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2}$ at the honest interfaces. Thus, we can build a simulator that responds to the honest and adversarial queries precisely as the simulator does in Game 9. The result is that the view in Game 10 is identical to the ideal world and it suffices to prove that any adjacent games are indistinguishable. Next we give the rigorous proof for the indistinguishably between each adjacent games.

Simulator In Ideal Game. Let $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ be the function pair that samples from the ideal OT family $\mathcal{T}_{\mathsf{EOT}}$, the simulator works as follows. In Game 10, the simulator in the ideal game maintains six tables in the same way as in Game 9 except that those table items being the responses of $\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2}$ are replaced by the responses of $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}$, $\mathsf{I.EOT}_{\mathsf{B}_2}$ respectively.

By definition, the simulator $\mathcal{S}$ now has access to $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2}$ at the honest interfaces.

And for the adversarial queries, $\mathcal{S}$ works the same as in Game 9, by just using the tables and querying the honest interfaces.

Now we prove the indistinguishability between any adjacent games.

Claim 1. Game Real $\approx$ Game 0.

Proof. Recalling that the only difference between Game Real and Game 0 is that, in Game 0 the simulator additionally maintains several tables that are completely hidden from the adversary, hence we have

$$\Pr[\text{Game Real} = 1] = \Pr[\text{Game } 0 = 1]$$

Claim 2. Game 0 $\approx$ Game 1.

Proof. In Game 1, the simulator maintains longer tables than in Game 0, and the simulator responds to part of the queries at the adversarial interfaces by using those tables and calling the honest interfaces. For the queries to the honest interfaces, the simulator responds by forwarding the calls and responses of the algorithms $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}$. Moreover, the items stored in those tables are always consistent with the real game oracles $H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ at adversarial interfaces. Hence, the response of adversarial queries by either the real game oracles $H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ (in Game 0) or by real game oracles plus honest interfaces and tables (in Game 1) are identical, which implies

$$\Pr[\text{Game } 0 = 1] = \Pr[\text{Game } 1 = 1]$$

Claim 3. Game 1 $\approx$ Game 2.

We note that, in order not to be distinguished by the adversary, the simulator's responses at adversarial interfaces should satisfy the consistency conditions below:

1. There exists no two $\mathrm{SK}_0, \mathrm{SK}_0'$ such that $\mathcal{S}^{\mathsf{asyKG}_0}(\mathrm{SK}_0) = \mathcal{S}^{\mathsf{asyKG}_0}(\mathrm{SK}_0')$;
2. There exists no two $\mathrm{SK}_1, \mathrm{SK}_1'$ such that $\mathcal{S}^{\mathsf{asyKG}_1}(\mathrm{SK}_1) = \mathcal{S}^{\mathsf{asyKG}_1}(\mathrm{SK}_1')$;
3. $\mathsf{LoR}_b(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)) = \mathcal{S}^{H_0}(\mathrm{SK}_0)$;
4. $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1) = \mathcal{S}^{\mathsf{asyKG}_1}(\mathrm{SK}_1)$;
5. $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b) = \mathcal{S}^{\mathsf{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1)$;
6. $\mathsf{LoR}_b(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b), \mathrm{SK}_1)) = \mathcal{S}^{\mathsf{asySHK}_1}(\mathrm{SK}_1, \mathrm{PK}_0)$;
7. $\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1), b) = \mathsf{LoR}_b(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{B}_2}(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b), \mathrm{SK}_1))$.
8. $\mathcal{S}^{\mathsf{asySHK}_0}(\mathrm{SK}_0, \mathrm{PK}_1) = \mathcal{S}^{\mathsf{asySHK}_1}(\mathrm{SK}_1, \mathrm{PK}_0)$ if and only if $\mathrm{PK}_1 = \mathcal{S}^{\mathsf{asyKG}_1}(\mathrm{SK}_1)$ and $\mathrm{PK}_0 = \mathcal{S}^{\mathsf{asyKG}_0}(\mathrm{SK}_0)$.

Proof. The only difference between Game 1 and Game 2 occurs in the Case 2, where $(\mathrm{SK}_0, b) \notin T_{H_0}$. In Game 1, the simulator responds to H_0 query (SK_0, b) with $H_0(\mathrm{SK}_0, b)$ while in Game 2, the simulator responds with $\mathsf{LoR}_b(\mathit{\Pi}.\mathsf{EOT}_{\mathsf{A}_1}(\tilde{\mathrm{SK}}_0))$.

Since the adversary knows nothing of $H_0(\mathrm{SK}_0, b)$, the distribution of $H_0(\mathrm{SK}_0, b)$ should be uniformly random in Dom_Q,and

$\mathsf{LoR}_b(\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b))$ has identical distribution. Due to definition, $H_0(\mathrm{SK}_0, b)$ and $\mathsf{LoR}_b(\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0))$ are identical. Besides, all the consistency conditions hold. Therefore, from the adversary's view, Game 1 and Game 2 are identical. Hence,

$$\Pr[\text{Game } 1 = 1] = \Pr[\text{Game } 2 = 1]$$

Claim 4. Game 2 $\approx$ Game 3.

Proof. Recalling that the only difference between Game 2 and Game 3 occurs in the Case 3 of simulating H_1.In Game 2, the simulator responds to a random query Q_d with $H_1(Q_d)$ while in Game 3, the simulator replaces it with a random string $\widetilde{\mathrm{PK}}$ in $\mathsf{DomAsy}_{\mathrm{PK}_0}$. Due to definition, the only case that the adversary queries Q_d to H_1 is when the adversary knows nothing of $H_1(Q_d)$. From the adversary's view, $H_1(Q_d)$ is uniformly distributed in $\mathsf{DomAsy}_{\mathrm{PK}_0}$, thus $\widetilde{\mathrm{PK}}$ is well-distributed. Since the responses of H_0, H_1 are independent and random strings, and for $\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ queries, the responses are identical in either game.

Therefore, from the adversary's view, Game 2 and Game 3 are indistinguishable except with negligible probability when $\widetilde{\mathrm{PK}}$ is already used in a previous query, which is bounded by $\frac{q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|}$. Hence

$$|\Pr[\text{Game } 2 = 1] - \Pr[\text{Game } 3 = 1]| \leq \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} \leq \mathsf{negl}(\lambda)$$

Claim 5. Game 3 $\approx$ Game 4.

Proof. Recalling that the only difference between Game 3 and Game 4 occurs in the Case 4 of simulating the responses of $\mathsf{I.asyKG}_0$ query. In Game 3, the simulator responds to a random query SK_0 with $\mathsf{I.asyKG}_0(\mathrm{SK}_0)$ while in Game 4, the simulator replaces it with a random string $\mathcal{S}^{H_1}(Q_1) \oplus \mathsf{LoR}_0(\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, 0))$. Due to definition, the only case that the adversary queries SK_0 to $\mathsf{I.asyKG}_0$ is when the adversary knows nothing of $\mathsf{I.asyKG}_0(\mathrm{SK}_0)$, which is uniformly distributed in $\mathsf{DomAsy}_{\mathrm{PK}_0}$ from the adversary's view. Since $\mathcal{S}^{H_1}(Q_1)$ is uniformly distributed in $\mathsf{DomAsy}_{\mathrm{PK}_0}$, $\mathcal{S}^{H_1}(Q_1) \oplus \mathsf{LoR}_0(\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, 0))$ is also well-distributed. And for $H_0, H_1, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ queries, the responses are identical in either game, both satisfying the consistency conditions.

Therefore, from the adversary's view, Game 3 and Game 4 are indistinguishable.

Claim 6. Game 4 $\approx$ Game 5.

Proof. Recalling that the only difference between Game 4 and Game 5 occurs in the Case 4 of simulating $\mathsf{I.asyKG}_1$.In Game 4, the simulator responds to a random query SK_0 with $\mathsf{I.asyKG}_1(\mathrm{SK}_0)$ while in Game 5, the simulator replaces it with $\Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$. Due to definition, the distributions of $\mathsf{I.asyKG}_1(\mathrm{SK}_0)$ and $\Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$ are identical. And for $H_0, H_1, \mathsf{I.asyKG}_0, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$

queries, the responses are identical in either game, both satisfying the consistency conditions.

Therefore, from the adversary's view, Game 4 and Game 5 are indistinguishable.

Claim 7. Game 5 $\approx$ Game 6.

Proof. Recalling that the only difference between Game 5 and Game 6 occurs in the Case 4 of simulating $\mathsf{I.asySHK_0}$.In Game 5, the simulator responds to a query $(\mathrm{SK}_0, \mathrm{PK}_1)$ with $\mathsf{I.asySHK_0}(\mathrm{SK}_0, \mathrm{PK}_1)$ while in Game 6, the simulator replaces it with $\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ for a random bit b. Due to definition, the distributions of $\mathsf{I.asySHK_0}(\mathrm{SK}_0, \mathrm{PK}_1)$ and $\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ are identical. And for $H_0, H_1, \mathsf{I.asyKG_0}, \mathsf{I.asyKG_1}, \mathsf{I.asySHK_1}$ queries, the responses are identical in either game, both satisfying the consistency conditions.

Therefore, from the adversary's view, Game 5 and Game 6 are indistinguishable.

Claim 8. Game 6 $\approx$ Game 7.

Proof. Recalling that the only difference between Game 6 and Game 7 occurs in the Case 7 of simulating $\mathsf{I.asySHK_1}$.In Game 6, the simulator responds to a random query $(\mathrm{SK}_1, \mathrm{PK}_0)$ with $\mathsf{I.asySHK_1}(\mathrm{SK}_1, \mathrm{PK}_0)$ while in Game 7, the simulator replaces it with a random string K uniformly distributed in $\mathsf{DomAsy}_\mathrm{K}$.

Due to definition, the only case that the adversary queries $(\mathrm{SK}_1, \mathrm{PK}_0)$ to $\mathsf{I.asySHK_1}$ is when the adversary knows nothing of $\mathsf{I.asySHK_1}(\mathrm{SK}_1, \mathrm{PK}_0)$, which is uniformly distributed in Dom_K from the adversary's view. Hence, K is well-distributed. And for $H_0, H_1, \mathsf{I.asyKG_0}, \mathsf{I.asyKG_1}, \mathsf{I.asySHK_0}$ queries, the responses are identical in either game, both satisfying the consistency conditions.

Therefore, from the adversary's view, Game 6 and Game 7 are indistinguishable unless collision event in Dom_K occurs, which is bounded by $\frac{q^2}{|\mathsf{Dom}_\mathrm{K}|}$. Hence we have

$$|\Pr[\text{Game } 6 = 1] - \Pr[\text{Game } 7 = 1]| \leq \frac{q^2}{|\mathsf{Dom}_\mathrm{K}|} \leq \mathsf{negl}(\lambda)$$

Claim 9. Game 7 $\approx$ Game 8.

Proof. Let $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ be the function pair that samples from $\mathcal{T}_{\mathsf{EOT}}$. We note that in Game 7, the simulator responds all of the adversarial interfaces just using tables and the algorithms $(\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$ at honest interfaces, it never directly calls the real oracles at honest interfaces. We immediately observe that $\mathcal{S}_8$ is identical to the simulator $\mathcal{S}$ in the ideal game, which refers to

$$|\Pr[\text{Game } 8 = 1] - \Pr[\text{Ideal Game} = 1]|$$

Therefore, it is rest to prove that Game 7 and Game 8 are close.

H_0, H_1 are random oracles, $\mathsf{I.asyKG}_0, \mathsf{I.asyKG}_1, \mathsf{I.asySHK}_0, \mathsf{I.asySHK}_1$ are random injections with the shared key property.

Conditioned on the oracles H_0, H_1 have no collisions, for any SK_0, b and SK_1, it's oblivious that the distributions of $\Pi.\mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)$ and $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)$ are identical, and the distributions of $\Pi.\mathsf{EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$ and $\mathsf{I.EOT}_{\mathsf{B}_1}(\mathrm{SK}_1)$ are identical; and for any $(\mathrm{SK}_0, \mathrm{PK}_1)$, the distributions of $\Pi.\mathsf{EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ and $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, \mathrm{PK}_1, b)$ are identical, and for any (SK_1, Q) the distributions of $\Pi.\mathsf{EOT}_{\mathsf{B}_2}(\mathrm{SK}_1, Q)$ and $\mathsf{I.EOT}_{\mathsf{B}_2}(\mathrm{SK}_1, Q)$ are identical.

That the oracle H_0 has collision means there are two queries SK_0 and SK_0' to H_0 such that $\mathrm{SK}_0 \neq \mathrm{SK}_0'$ and $H_0(\mathrm{SK}_0) = H_0(\mathrm{SK}_0')$. That the oracle H_1 has collision means there are two queries Q_d and Q_d' to H_1 such that $Q_d \neq Q_d'$ and $H_1(Q_d) = H_1(Q_d')$;

If none of the collision occurs, we can replace $(\Pi.\mathsf{EOT}_{\mathsf{A}_1}, \Pi.\mathsf{EOT}_{\mathsf{A}_2}, \Pi.\mathsf{EOT}_{\mathsf{B}_1}, \Pi.\mathsf{EOT}_{\mathsf{B}_2})$ with $(\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$, which represents Game 8. Moreover, we can bound the probability of H_0, H_1 collision by

$$\Pr[\mathsf{Collision}] \leq \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} \leq \mathsf{negl}(\lambda),$$

which refers to

$$|\Pr[\text{Game } 7 = 1] - \Pr[\text{Game } 8 = 1]| \leq \Pr[\mathsf{Collision}] \leq \mathsf{negl}(\lambda)$$

Combining all claims together, we have

$$|\Pr[\text{Real Game} = 1] - \Pr[\text{Ideal Game} = 1]| \leq \frac{q^2}{|\mathsf{DomAsy}_{\mathrm{SK}_0}|} + \frac{2q^2}{|\mathsf{DomAsy}_{\mathrm{PK}_0}|} + \frac{q^2}{|\mathsf{Dom}_{\mathrm{K}}|} \leq \mathsf{negl}(\lambda),$$

thus we complete the entire proof.

C Proof of Theorem 3

Proof. According to the definition of indifferentiability, in the real world, the differentiator $\mathcal{D}$ has oracle access to $(\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$ via the honest interface and oracle access to $(H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ via the adversarial interface.

In contrast, in the ideal world, the differentiator $\mathcal{D}$ has oracle access to $(\mathsf{I.symKG}, \mathsf{I.symSHK})$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface. Therefore, to establish a proof, we need to build an explicit (and efficient) simulator $\mathcal{S}$ that simulates the rest oracles $(H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2})$ properly by making queries to $(\mathsf{I.symKG}, \mathsf{I.symSHK})$. Namely, for any PPT differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real game is computationally close to the view in the ideal game. To do so, we will go through with a sequence of hybrid games, where in each game, the simulator responds to all of the queries (both honest and adversarial) in a slightly different way and the last game is the same as the ideal world. Note that the differentiator $\mathcal{D}$ can make at most q queries to the oracles, where $q = \mathrm{poly}(\lambda)$.

The simulator in the ideal world works as follows. The simulator $\mathcal{S}$ has the external oracle access to an ideal symmetric NIKE scheme $\mathsf{I.symNIKE} = (\mathsf{I.symKG}, \mathsf{I.symSHK})$; and the simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$.

Simulator $\mathcal{S}$

$\mathcal{S}^{H_1}(\text{SK})$:

Case 1: if $\exists(\text{SK}, \overline{\text{SK}}) \in T_{H_1}$,
 return $\overline{\text{SK}}$;
Case 2: if $\exists(\text{SK}, 0, \text{PK}^L) \in T_{\text{OT}_{A_1}} \wedge (\overline{\text{SK}}, \text{PK}^R) \in T_{\text{OT}_{B_1}}$, s.t. $\mathsf{I.symKG}(\text{SK}) = \text{PK}^L \| \text{PK}^R$,
 return $\overline{\text{SK}}$;
Case 3: if $\exists(\text{SK}, 0, \text{PK}_1^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_0, \text{PK}_1^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\mathsf{I.symSHK}(\text{SK}, \text{PK}_1^L \| \text{PK}_1^R) = K$,
 return $\overline{\text{SK}}_0$;
Case 4: if $\exists(\text{SK}, 0, \text{PK}_0^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\mathsf{I.symSHK}(\text{SK}, \text{PK}_0^L \| \text{PK}_0^R) = K$,
 return $\overline{\text{SK}}_1$;
Case 5: Otherwise, sample $\overline{\text{SK}} \leftarrow \{0,1\}^{\ell_1(\lambda)}$;$T_{H_1} = T_{H_1} \cup (\text{SK}, \overline{\text{SK}})$;
 return $\overline{\text{SK}}$.

$\mathcal{S}^{\mathsf{EOT_{A_1}}}(\text{SK}, b)$:

Case 1: if $\exists(\text{SK}, b, \text{PK}^L) \in T_{\text{OT}_{A_1}}$,
 return PK^L;
If $b = 0$:
Case 2: if $\exists(\text{SK}, 0, \text{PK}_1^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\overline{\text{SK}}_1, \text{PK}_1^R) \in T_{\text{OT}_{B_1}}$ s.t. $K_A = K_B$,
 return PK_0^L;
Case 3: if $\exists(\text{SK}, 0, \text{PK}_0^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_0, \text{PK}_1^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\overline{\text{SK}}_0, \text{PK}_0^R) \in T_{\text{OT}_{B_1}}$ s.t. $K_A = K_B$,
 return PK_1^L;
Case 4: otherwise, query $\mathsf{I.symKG}$ with SK, obtain PK (which has $\ell_2(\lambda)$ bits), truncate the first $2\ell_5(\lambda)$ bits as PK^L;
 $T_{\text{OT}_{A_1}} = T_{\text{OT}_{A_1}} \cup (\text{SK}, 0, \text{PK}^L)$;
 return PK^L;
If $b = 1$:
Case 5: sample $\text{PK}^L \leftarrow \{0,1\}^{2\ell_5(\lambda)}$; $T_{\text{OT}_{A_1}} = T_{\text{OT}_{A_1}} \cup (\text{SK}, 1, \text{PK}^L)$;
 return PK^L.

$\mathcal{S}^{\mathsf{EOT_{A_2}}}(\text{SK}, b, \widetilde{\text{PK}}^R)$:

Case 1: if $\exists(\text{SK}, b, \widetilde{\text{PK}}^R, K_A) \in T_{\text{OT}_{A_2}}$,
 return K_A;
If $b = 0$:
Case 2: if $\exists(\text{SK}, 0, \text{PK}_0^L) \in T_{\text{OT}_{A_1}} \wedge (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\overline{\text{SK}}_1, \widetilde{\text{PK}}^R) \in$

$T_{\mathrm{OT}_{B_1}}$,
return K_B;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_0, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}}$,
return K_B;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$, s.t. $\mathsf{I.symSHK}(\mathrm{SK}, \widetilde{\mathrm{PK}}^L \| \widetilde{\mathrm{PK}}^R) = K$,
return K_A;
Case 5: otherwise, sample $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A)$;
return K_A;
If $b = 1$:
Case 6: if $\exists(\overline{\mathrm{SK}}_1, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}, 1, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
return $\tilde{K_B}$.
Case 7: otherwise, sample $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup (\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^R, K_A)$;
return K_A.

$\mathcal{S}^{\mathsf{EOT_{B_1}}}(\overline{\mathrm{SK}})$:

Case 1: if $\exists(\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}}$,
return PK^R;
Case 2: if $\exists(\mathrm{SK}_1, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_1^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_1, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
return PK_0^R;
Case 3: if $\exists(\mathrm{SK}_0, 0, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_0^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_0, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
return PK_1^R;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1}$,
query $\mathsf{I.symKG}$ with SK, obtain PK ($\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits as PK^R;
return PK^R.
Case 5: otherwise, sample $\mathrm{SK} \leftarrow \{0,1\}^{\ell_1(\lambda)}$, query $\mathsf{I.symKG}$ with SK, obtain PK (which has $\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits as PK^R;
$T_{H_1} = T_{H_1} \cup (\mathrm{SK}, \overline{\mathrm{SK}})$,$T_{\mathrm{OT}_{B_1}} = T_{\mathrm{OT}_{B_1}} \cup (\overline{\mathrm{SK}}, \mathrm{PK}^R)$;
return PK^R.

$\mathcal{S}^{\mathsf{EOT_{B_2}}}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)$:

Case 1: if $\exists(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
return $K_B, \tilde{K_B}$;
Case 2: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$, s.t. $\mathsf{I.symSHK}(\mathrm{SK}, \widetilde{\mathrm{PK}}^L \| \widetilde{\mathrm{PK}}^R) = K$,
sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$,
return $K_B, \tilde{K_B}$;
Case 3: if $\exists(\mathrm{SK}', 0, \widetilde{\mathrm{PK}}^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}', 0, \mathrm{PK}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$,

sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$,
return $K_A, \tilde{K_B}$;
Case 4: if $\exists(\text{SK}', 1, \widetilde{\text{PK}}^L) \in T_{\text{OT}_{A_1}} \wedge (\overline{\text{SK}}, \text{PK}^R) \in T_{\text{OT}_{B_1}} \wedge (\text{SK}', 1, \text{PK}^R, K_A) \in T_{\text{OT}_{A_2}}$,
sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$,
return $\tilde{K_B}, K_A$;
Case 5: otherwise, sample $K_B, \tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\text{OT}_{B_2}} = T_{\text{OT}_{B_2}} \cup (\overline{\text{SK}}, \widetilde{\text{PK}}^L, K_B, \tilde{K_B})$;
return $K_B, \tilde{K_B}$.

$\underline{\mathcal{S}^{H_0}(K_A, K_B)}$:
Case 1: if $\exists(K_A, K_B, K) \in T_{H_0}$,
return K;
Case 2: if $\exists(\text{SK}_0, 0, \text{PK}_1^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_0, \text{PK}_1^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\text{SK}_0, \overline{\text{SK}}_0) \in T_{H_1}$,
query $\mathsf{I.symSHK}$ with $\text{SK}_0, \text{PK}_1^L \| \text{PK}_1^R$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K;
Case 3: if $\exists(\text{SK}_1, 0, \text{PK}_0^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\text{SK}_1, \overline{\text{SK}}_1) \in T_{H_1}$,
query $\mathsf{I.symSHK}$ with $(\text{SK}_1, \text{PK}_0^L \| \text{PK}_0^R)$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K;
Case 4: otherwise, $K \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K.

Next, we describe our simulator $\mathcal{S}$ in the hybrid games.

Game 0. This game is identical to the real game except that the simulator maintains six tables for the adversarial interfaces, referring to H_0 table, H_1 table, OT_{A_1} table, OT_{A_2} table, OT_{B_1} table and OT_{B_2} table. The tables are denoted as T_{H_0}, T_{H_1}, $T_{\text{OT}_{A_1}}$, $T_{\text{OT}_{A_2}}$, $T_{\text{OT}_{B_1}}$, $T_{\text{OT}_{B_2}}$ respectively, in the following forms:

- $T_{H_1} := (\text{SK}_0, \overline{\text{SK}}_0)$ or $T_{H_1} := (\text{SK}_1, \overline{\text{SK}}_1)$;
- $T_{\text{OT}_{A_1}} := (\text{SK}_0, b, \text{PK}_0^L)$, or $T_{\text{OT}_{A_1}} := (\text{SK}_1, b, \text{PK}_1^L)$;
- $T_{\text{OT}_{A_2}} := (\text{SK}_0, b, \text{PK}_1^R, K_A)$ or $T_{\text{OT}_{A_2}} := (\text{SK}_1, b, \text{PK}_0^R, K_A)$;
- $T_{\text{OT}_{B_1}} := (\overline{\text{SK}}_0, \text{PK}_0^R)$ or $T_{\text{OT}_{B_1}} := (\overline{\text{SK}}_1, \text{PK}_1^R)$;
- $T_{\text{OT}_{B_2}} := (\overline{\text{SK}}_0, \text{PK}_1^L, K_B, \tilde{K_B})$ or $T_{\text{OT}_{B_2}} := (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B})$;
- $T_{H_0} := (K_A, K_B, K)$;

Concretely, the simulator responds to the queries by forwarding the responses of the corresponding oracles, such that the simulator's responses are the same as in the real world. For instance, $\mathcal{S}_0^{H_1}(\text{SK}) = H_1(\text{SK})$, $\mathcal{S}_0^{H_0}(K_A, K_B) = H_0(K_A, K_B)$, $\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}, b) = \mathsf{I.EOT}_{\mathsf{A}_1}(\text{SK}, b)$, $\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\text{SK}}) = \mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\text{SK}})$, $\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{A}_2}}(\text{SK}', b, \text{PK}^R) = \mathsf{I.EOT}_{\mathsf{A}_2}(\text{SK}', b, \text{PK}^R)$, $\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{B}_2}}(\overline{\text{SK}}, \widetilde{\text{PK}}^L) = \mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\text{SK}}, \widetilde{\text{PK}}^L)$ and so forth.

Simulator $\mathcal{S}_0$

$\underline{\mathcal{S}_0^{H_1}(\mathrm{SK})}$:

$\overline{\mathrm{SK}} \leftarrow H_1(\mathrm{SK})$, return $\overline{\mathrm{SK}}$.

$\underline{\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{A}_1}}(\mathrm{SK}, b)}$:

$\mathrm{PK}^L \leftarrow \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, b)$, return PK^L.

$\underline{\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{A}_2}}(\mathrm{SK}_0, b, \mathrm{PK}_1^R)}$:

$K_A \leftarrow \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, b, \mathrm{PK}_1^R)$, return K_A.

$\underline{\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\mathrm{SK}})}$:

$\mathrm{PK}^R \leftarrow \mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$, return PK^R.

$\underline{\mathcal{S}_0^{\mathsf{EOT}_{\mathsf{B}_2}}(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L)}$:

$K_B \leftarrow \mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L)$, return K_B.

$\underline{\mathcal{S}_0^{H_0}(K_A, K_B)}$:

$K \leftarrow H_0(K_A, K_B)$, return K.

For the tables, the simulator maintains them as follows.

1. H_1-table T_{H_1}: initially empty, consists of tuples with form of $(\mathrm{SK}, \overline{\mathrm{SK}})$. Once the adversary queries oracle H_1 with SK which does not exist in T_{H_1}, the simulator inserts $(\mathrm{SK}, H_1(\mathrm{SK}))$ into the T_{H_1}-table.
2. $\mathsf{I.EOT}_{\mathsf{A}_1}$-table $T_{\mathrm{OT}_{A_1}}$: initially empty, consists of tuples with form of $(\mathrm{SK}, b, \mathrm{PK}^L)$. Once the adversary queries $\mathsf{I.EOT}_{\mathsf{A}_1}$ with (SK, b) which does not exist in $T_{\mathrm{OT}_{A_1}}$-table, the simulator inserts $(\mathrm{SK}, b, \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, b)$ into the $T_{\mathrm{OT}_{A_1}}$-table.
3. $\mathsf{I.EOT}_{\mathsf{A}_2}$-table $T_{\mathrm{OT}_{A_2}}$: initially empty, consists of tuples with form of $(\mathrm{SK}_0, b, \mathrm{PK}_1^R, K_A)$ or $(\mathrm{SK}_1, b, \mathrm{PK}_0^R, K_A)$. Once the adversary queries $\mathsf{I.EOT}_{\mathsf{A}_2}$ with $(\mathrm{SK}_0, b, \mathrm{PK}_1^R)$ or $(\mathrm{SK}_1, b, \mathrm{PK}_0^R)$ which does not exist in $T_{\mathrm{OT}_{A_2}}$-table, the simulator inserts $(\mathrm{SK}_0, b, \mathrm{PK}_1^R, \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, b, \mathrm{PK}_1^R)$ or $(\mathrm{SK}_1, b, \mathrm{PK}_0^R, \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_1, b, \mathrm{PK}_0^R)$ into the $T_{\mathrm{OT}_{A_2}}$-table.
4. $\mathsf{I.EOT}_{\mathsf{B}_1}$-table $T_{\mathrm{OT}_{B_1}}$: initially empty, consists of tuples with form of $(\overline{\mathrm{SK}}, \mathrm{PK}^R)$. Once the adversary queries $\mathsf{I.EOT}_{\mathsf{B}_1}$ with $\overline{\mathrm{SK}}$ which does not exist in the $T_{\mathrm{OT}_{B_1}}$-table, the simulator inserts $(\overline{\mathrm{SK}}, \mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}}))$ into the $T_{\mathrm{OT}_{B_1}}$-table.
5. $\mathsf{I.EOT}_{\mathsf{B}_2}$-table $T_{\mathrm{OT}_{B_2}}$: initially empty, consists of tuples with form of $(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B})$ (or $(\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B})$). Once the adversary queries $\mathsf{I.EOT}_{\mathsf{B}_2}$ with $(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L)$ (or $(\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L)$) which does not exist in $T_{\mathrm{OT}_{B_2}}$-table, the simulator inserts $(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, \mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L)$ (or $(\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, \mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L)$) into the $T_{\mathrm{OT}_{B_2}}$-table.

6. H_0-table T_{H_0}: initially empty, consists of tuples with form of (K_A, K_B, K). Once the adversary queries oracle H_0 with K_A, K_B which does not exist in T_{H_0}, the simulator inserts $(K_A, K_B, H_0(K_A, K_B))$ into the T_{H_0} table.

At this point all the queries are responded by the real oracles, and these tables are just keeping track of information related to $\mathcal{D}$'s queries (to the adversarial interfaces) and completely hidden to the adversary, hence the adversary's view in real game is identical to the one in Game 0.

Next, we illustrate an alternative way to answer part of the queries, by using these tables and the honest interfaces.

Simulator $\mathcal{S}_1$

$\underline{\mathcal{S}_1^{H_1}(\mathrm{SK})}$:

Case 1: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1}$,
  return $\overline{\mathrm{SK}}$;
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}}$, s.t. $\Pi.\mathsf{symKG}(\mathrm{SK}) = \mathrm{PK}^L \| \mathrm{PK}^R$,
  return $\overline{\mathrm{SK}}$;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \mathrm{PK}_1^L \| \mathrm{PK}_1^R) = K$,
  return $\overline{\mathrm{SK}}_0$;
Case 4: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \mathrm{PK}_0^L \| \mathrm{PK}_0^R) = K$,
  return $\overline{\mathrm{SK}}_1$;
Case 5: Otherwise, $\overline{\mathrm{SK}} = H_1(\mathrm{SK})$; $T_{H_1} = T_{H_1} \cup (\mathrm{SK}, \overline{\mathrm{SK}})$; return $\overline{\mathrm{SK}}$.

$\underline{\mathcal{S}_1^{\mathsf{EOT}_{\mathsf{A}_1}}(\mathrm{SK}, b)}$:

Case 1: if $\exists(\mathrm{SK}, b, \mathrm{PK}^L) \in T_{\mathrm{OT}_{A_1}}$,
  return PK^L;
If $b = 0$**:**
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_1^R) \in T_{\mathrm{OT}_{B_1}}$ s.t. $K_A = K_B$,
  return PK_0^L;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_0^R) \in T_{\mathrm{OT}_{B_1}}$ s.t. $K_A = K_B$,
  return PK_1^L;
Case 4: otherwise, $\mathrm{PK}^L = \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 0)$; $T_{\mathrm{OT}_{A_1}} = T_{\mathrm{OT}_{A_1}} \cup (\mathrm{SK}, 0, \mathrm{PK}^L)$;
  return PK^L;
If $b = 1$**:**
Case 5: $\mathrm{PK}^L = \mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 0)$; $T_{\mathrm{OT}_{A_1}} = T_{\mathrm{OT}_{A_1}} \cup (\mathrm{SK}, 0, \mathrm{PK}^L)$;
  return PK^L;

$\underline{\mathcal{S}_1^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\mathrm{SK}})}$:

Case 1: if $\exists(\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}}$,

return PK^R;
Case 2: if $\exists(\mathrm{SK}_1, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_1^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_1, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
return PK_0^R;
Case 3: if $\exists(\mathrm{SK}_0, 0, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_0^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_0, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
return PK_1^R;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1}$,
query $\Pi.\mathsf{symKG}$ with SK, obtain PK ($\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits as PK^R;
return PK^R.
Case 5: otherwise, $\mathrm{PK}^R = \mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$; $T_{\mathrm{OT}_{B_1}} = T_{\mathrm{OT}_{B_1}} \cup (\overline{\mathrm{SK}}, \mathrm{PK}^R)$;
return PK^R.

$\underline{\mathcal{S}_1^{\mathsf{EOT}_{\mathsf{A}_2}}(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R):}$

Case 1: if $\exists(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$,
return K_A;
If $b = 0$:
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_1, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}}$,
return K_B;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_0, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}}$,
return K_B;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \mathrm{PK}^L \| \widetilde{\mathrm{PK}}^R) = K$,
return K_A;
Case 5: otherwise, $K_A = \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R)$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A)$;
return K_A;
If $b = 1$:
Case 6: if $\exists(\overline{\mathrm{SK}}_1, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}, 1, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
return $\tilde{K_B}$.
Case 7: otherwise, $K_A = \mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^R)$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup (\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^R, K_A)$;
return K_A.

$\underline{\mathcal{S}_1^{\mathsf{EOT}_{\mathsf{B}_2}}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L):}$

Case 1: if $\exists(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
return $K_B, \tilde{K_B}$;
Case 2: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \widetilde{\mathrm{PK}}^L \| \widetilde{\mathrm{PK}}^R) = K$,
sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{B_2}} = T_{\mathrm{OT}_{B_2}} \cup (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B})$;

return $K_B, \tilde{K_B}$;
Case 3: if $\exists(\text{SK}', 0, \widetilde{\text{PK}}^L) \in T_{\text{OT}_{A_1}} \wedge (\overline{\text{SK}}, \text{PK}^R) \in T_{\text{OT}_{B_1}} \wedge (\text{SK}', 0, \text{PK}^R, K_A) \in T_{\text{OT}_{A_2}}$,
sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\text{OT}_{B_2}} = T_{\text{OT}_{B_2}} \cup (\overline{\text{SK}}, \widetilde{\text{PK}}^L, K_A, \tilde{K_B})$;
return $K_A, \tilde{K_B}$;
Case 4: if $\exists(\text{SK}', 1, \widetilde{\text{PK}}^L) \in T_{\text{OT}_{A_1}} \wedge (\overline{\text{SK}}, \text{PK}^R) \in T_{\text{OT}_{B_1}} \wedge (\text{SK}', 1, \text{PK}^R, K_A) \in T_{\text{OT}_{A_2}}$,
sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\text{OT}_{B_2}} = T_{\text{OT}_{B_2}} \cup (\overline{\text{SK}}, \widetilde{\text{PK}}^L, \tilde{K_B}, K_A)$;
return $\tilde{K_B}, K_A$;
Case 5: otherwise, $(K_B, \tilde{K_B}) = \mathsf{I.EOT_{B_2}}(\overline{\text{SK}}, \widetilde{\text{PK}}^L)$; $T_{\text{OT}_{B_2}} = T_{\text{OT}_{B_2}} \cup (\overline{\text{SK}}, \widetilde{\text{PK}}^L, K_B, \tilde{K_B})$;
return $K_B, \tilde{K_B}$.

$\underline{\mathcal{S}_1^{H_0}(K_A, K_B)}$:
Case 1: if $\exists(K_A, K_B, K) \in T_{H_0}$,
return K;
Case 2: if $\exists(\text{SK}_0, 0, \text{PK}_1^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_0, \text{PK}_1^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\text{SK}_0, \overline{\text{SK}}_0) \in T_{H_1}$,
query $\mathit{\Pi}.\mathsf{symSHK}$ with $\text{SK}_0, \text{PK}_1^L \| \text{PK}_1^R$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K;
Case 3: if $\exists(\text{SK}_1, 0, \text{PK}_0^R, K_A) \in T_{\text{OT}_{A_2}} \wedge (\overline{\text{SK}}_1, \text{PK}_0^L, K_B, \tilde{K_B}) \in T_{\text{OT}_{B_2}} \wedge (\text{SK}_1, \overline{\text{SK}}_1) \in T_{H_1}$,
query $\mathit{\Pi}.\mathsf{symSHK}$ with $(\text{SK}_1, \text{PK}_0^L \| \text{PK}_0^R)$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K;
Case 4: otherwise, $K = H_0(K_A, K_B)$; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
return K.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries at adversarial interfaces. The simulator $\mathcal{S}_1$ has the external oracle access to the symmetric NIKE scheme $\mathit{\Pi}.\mathsf{symNIKE} = (\mathit{\Pi}.\mathsf{symKG}, \mathit{\Pi}.\mathsf{symSHK})$. The simulator $\mathcal{S}_1$ will provide the interfaces for the external differentiator $\mathcal{D}$. Specifically, the simulator responds to the oracles as in the box of $\mathcal{S}_1$.

Compared to Game 0, in Game 1 the simulator keeps a longer table, and for part of the queries, the simulator responds to them in an alternative way, which is only using the tables and the honest interfaces. Note that, for the $\mathsf{I.EOT_{B_2}}$ queries, when the useful half of the output is determined using the tables and honest interfaces, the other half of the output (which does not influence all other queries) is sampled uniformly at random, without influencing the distribution of the overall distribution. Moreover, in Game 1, the tuples stored in the tables correspond to the response by the real oracles to the adversary's queries, except for the Case 2, Case 3 and Case 4 of $\mathsf{I.EOT_{B_2}}$ queries, the responses for which consists of two parts with one part being the same in Game 0 and Game 1 and the other part having identical distribution in Game 0 and Game 1. Note that the other part (in most cases the second part) has no influence on all the other oracles/ interfaces. identical distribution with the responses of real oracles.

Hence, in Game 0 and Game 1, the responses of any query other than $\mathsf{I.EOT}_{\mathsf{B}_2}$ query are identical, and the responses for $\mathsf{I.EOT}_{\mathsf{B}_2}$ query have statistically close distribution. Therefore, the adversary's view in Game 1 is indistinguishable from the view in Game 0. However, in Game 1, the simulator can only answer part of the queries by tables and honest interfaces, and for the rest it has to call the real oracles. Thus, in the following hybrid games, we will illustrate additional alternative ways to respond to the rest queries, without changing the view significantly.

Game 2. This game is identical to Game 1, except for responding to H_1 queries. The simulator responds to a query SK on H_1 as follows:

Simulator $\mathcal{S}_2$

$\underline{\mathcal{S}_2^{H_1}(\mathrm{SK})}$:

Case 1: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1}$,
  return $\overline{\mathrm{SK}}$;
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}}$, s.t.
  $\Pi.\mathsf{symKG}(\mathrm{SK}) = \mathrm{PK}^L \| \mathrm{PK}^R$,
  return $\overline{\mathrm{SK}}$;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \mathrm{PK}_1^L \| \mathrm{PK}_1^R) = K$,
  return $\overline{\mathrm{SK}}_0$;
Case 4: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (K_A, K_B, K) \in T_{H_0}$, s.t. $\Pi.\mathsf{symSHK}(\mathrm{SK}, \mathrm{PK}_0^L \| \mathrm{PK}_0^R) = K$,
  return $\overline{\mathrm{SK}}_1$;
Case 5: otherwise, sample $\overline{\mathrm{SK}} \leftarrow \{0,1\}^{\ell_3(\lambda)}$; $T_{H_1} = T_{H_1} \cup (\mathrm{SK}, \overline{\mathrm{SK}})$;
  return $\overline{\mathrm{SK}}$.

The only difference between Game 1 and Game 2 occurs in the Case 5 of H_1 queries where the corresponding $\overline{\mathrm{SK}}$ never appears in the tables maintained by the simulator. In Game 1, the simulator responds with $H_1(\mathrm{SK})$; while in Game 2, the simulator responds with a random string $\overline{\mathrm{SK}}$ in $\{0,1\}^{\ell_3(\lambda)}$. Due to definition, the only case that the adversary queries H_1 with such SK is when the adversary $\mathcal{D}$ knows nothing of $H_1(\mathrm{SK})$. Therefore, from $\mathcal{D}$'s view, $H_1(\mathrm{SK})$ is uniformly distributed in $\{0,1\}^{\ell_3(\lambda)}$, and $\overline{\mathrm{SK}}$ has the same distribution, which implies $\mathcal{D}$'s view in Game 2 are indistinguishable from its view in Game 1.

Game 3. This game is identical to Game 2, except for responding to $\mathsf{I.EOT}_{\mathsf{A}_1}$ queries. The simulator responds to a query (SK, b) as follows:

Simulator $\mathcal{S}_3$

$\underline{\mathcal{S}_3^{\mathsf{EOT}_{\mathsf{A}_1}}(\mathrm{SK}, b)}$:

Case 1: if $\exists(\mathrm{SK}, b, \mathrm{PK}^L) \in T_{\mathrm{OT}_{A_1}}$,
 return PK^L;
If $b = 0$**:**
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_1^R) \in T_{\mathrm{OT}_{B_1}}$ s.t. $K_A = K_B$,
 return PK_0^L;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_0^R) \in T_{\mathrm{OT}_{B_1}}$ s.t. $K_A = K_B$,
 return PK_1^L;
Case 4: otherwise, query $\Pi.\mathsf{symKG}$ with SK, obtain PK (which has $\ell_2(\lambda)$ bits), truncate its first $2\ell_5(\lambda)$ bits as PK^L;
 $T_{\mathrm{OT}_{A_1}} = T_{\mathrm{OT}_{A_1}} \cup (\mathrm{SK}, 0, \mathrm{PK}^L)$;
 return PK^L;
If $b = 1$**:**
Case 5: sample $\tilde{\mathrm{PK}}^L \leftarrow \{0,1\}^{2\ell_5(\lambda)}$; $T_{\mathrm{OT}_{A_1}} = T_{\mathrm{OT}_{A_1}} \cup (\mathrm{SK}, 1, \tilde{\mathrm{PK}}^L)$;
 return $\tilde{\mathrm{PK}}^L$.

The only difference between Game 2 and Game 3 occurs in the Case 4 and Case 5 of $\mathsf{I.EOT}_{\mathsf{A}_1}$ query.

For Case 4, in Game 2, the simulator responds to a query $(\mathrm{SK}, 0)$ with $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 0)$; while in Game 3, the simulator responds with PK^L which is the first $2\ell_5(\lambda)$ bits of $\Pi.\mathsf{symKG}(\mathrm{SK})$. Due to definition, $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 0)$ and PK^L are identical. For Case 5, in Game 2, the simulator responds to a query $(\mathrm{SK}, 1)$ with $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$; while in Game 3, the simulator responds with a uniformly sampled random string $\tilde{\mathrm{PK}}^L \leftarrow \{0,1\}^{2\ell_5(\lambda)}$. Due to definition, $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$ is uniformly distributed in $\{0,1\}^{2\ell_5(\lambda)}$, thus the distributions of $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$ and $\tilde{\mathrm{PK}}^L$ are identical. Therefore, the adversary's views in Game 3 and Game 2 are indistinguishable.

Game 4. This game is identical to Game 3, except for responding to $\mathsf{I.EOT}_{\mathsf{B}_1}$ queries. The simulator responds to a query $\overline{\mathrm{SK}}$ on $\mathsf{I.EOT}_{\mathsf{B}_1}$ as follows:

Simulator $\mathcal{S}_4$

$\underline{\mathcal{S}_4^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\mathrm{SK}})}$:

Case 1: if $\exists(\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}}$,
 return PK^R;
Case 2: if $\exists(\mathrm{SK}_1, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_1^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_1, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
 return PK_0^R;
Case 3: if $\exists(\mathrm{SK}_0, 0, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}_0^L, K_B) \in T_{\mathrm{OT}_{B_2}} \wedge$

$(\mathrm{SK}_0, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}}$ s.t. $K_A = K_B$,
  return PK_1^R;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1}$,
query Π.symKG with SK, obtain PK (with $\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits as PK^R;
  return PK^R.
Case 5: otherwise, sample $\mathrm{SK} \leftarrow \{0,1\}^{\ell_1(\lambda)}$, query Π.symKG with SK, obtain PK (which has $\ell_2(\lambda)$ bits), truncate the last $\ell_4(\lambda)$ bits as PK^R;
  $T_{H_1} = T_{H_1} \cup (\mathrm{SK}, \overline{\mathrm{SK}})$, $T_{\mathrm{OT}_{B_1}} = T_{\mathrm{OT}_{B_1}} \cup (\overline{\mathrm{SK}}, \mathrm{PK}^R)$;
  return PK^R.

The only difference between Game 3 and Game 4 occurs in the Case 5.

In Game 3, the simulator responds to a $\mathsf{I.EOT}_{\mathsf{B}_1}$ query $\overline{\mathrm{SK}}$ with $\mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$; while in Game 4, the simulator responds with a string PK^R that is the last $\ell_4(\lambda)$ bits of Π.symKG(SK) for a random string SK in $\{0,1\}^{\ell_1(\lambda)}$. Due to definition, the only case that the adversary queries $\mathsf{I.EOT}_{\mathsf{B}_1}$ with $\overline{\mathrm{SK}}$ is when the adversary knows nothing of $\mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$. Therefore, from the adversary's view, $\mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$ is uniformly distributed in $\{0,1\}^{\ell_4(\lambda)}$. Since PK^R is also uniformly distributed in $\{0,1\}^{\ell_4(\lambda)}$, $\mathcal{D}$'s view in Game 4 and its view in Game 3 are indistinguishable.

Simulator $\mathcal{S}_5$

$\mathcal{S}_5^{\mathsf{EOT}_{\mathsf{A}_2}}(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R)$:

Case 1: if $\exists(\mathrm{SK}, b, \widetilde{\mathrm{PK}}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$,
  return K_A;
If $b = 0$**:**
Case 2: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_1, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}}$,
  return K_B;
Case 3: if $\exists(\mathrm{SK}, 0, \mathrm{PK}_1^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\overline{\mathrm{SK}}_0, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}}$,
  return K_B;
Case 4: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$, s.t. Π.symSHK$(\mathrm{SK}, \mathrm{PK}^L \| \widetilde{\mathrm{PK}}^R) = K$,
  return K_A;
Case 5: otherwise, uniformly sample $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A)$;
  return K_A;
If $b = 1$**:**
Case 6: if $\exists(\overline{\mathrm{SK}}_1, \widetilde{\mathrm{PK}}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}, 1, \mathrm{PK}_0^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
  return $\tilde{K_B}$.
Case 7: otherwise, uniformly sample $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{A_2}} = T_{\mathrm{OT}_{A_2}} \cup$

$(\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^{R}, K_A)$;
return K_A.

Game 5. This game is identical to Game 4, except for responding to $\mathsf{I.EOT}_{\mathsf{A}_2}$ queries. The simulator responds to a query $\mathrm{SK}_0, b, \mathrm{PK}_1^R$ on $\mathsf{I.EOT}_{\mathsf{A}_2}$ as in the box of Simulator $\mathcal{S}_5$.

The only difference between Game 4 and Game 5 occurs in the Case 5 and Case 7 of $\mathsf{I.EOT}_{\mathsf{A}_2}$ query. For the Case 5, in Game 4, the simulator responds to a $\mathsf{I.EOT}_{\mathsf{A}_2}$ query $(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^{R})$ with $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^{R})$; while in Game 5, the simulator responds with a randomly sampled string $K_A \leftarrow \{0,1\}^{n_3(\lambda)}$. Since the only case that the adversary $\mathcal{D}$ queries $(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^{R})$ is when $\mathcal{D}$ knows nothing of $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^{R})$, which refers to $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^{R})$ is uniformly distributed in $\{0,1\}^{n_3(\lambda)}$. Hence, the distributions of $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, 0, \mathrm{PK}_1^R)$ and K_A are identical. For the Case 7, in Game 4, the simulator responds to a $\mathsf{I.EOT}_{\mathsf{A}_2}$ query $(\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^{R})$ with $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^{R})$; while in Game 5, the simulator responds with a random string $\tilde{K} \leftarrow \{0,1\}^{n_3(\lambda)}$. Similar to the above analysis, $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}, 1, \widetilde{\mathrm{PK}}^{R})$ and $\tilde{K}$ are both uniformly distributed in $\{0,1\}^{n_3(\lambda)}$. Therefore, the adversary $\mathcal{D}$'s view in Game 5 and its view in Game 4 are indistinguishable.

Game 6. This game is identical to Game 5, except for responding to $\mathsf{I.EOT}_{\mathsf{B}_2}$ queries. The simulator responds to a query $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^{L})$ on $\mathsf{I.EOT}_{\mathsf{B}_2}$ as in the box of Simulator $\mathcal{S}_6$.

The only difference between Game 5 and Game 6 occurs in the Case 5. In Game 5, the simulator responds to a $\mathsf{I.EOT}_{\mathsf{B}_2}$ query $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^{L})$ with $\mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^{L})$; while in Game 6, the simulator responds with a random string $K_B \| \tilde{K}_B$.

Due to definition, from the adversary $\mathcal{D}$'s view, $\mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^{L})$ is uniformly distributed in $\{0,1\}^{n_3(\lambda)}$, hence $K_B \| \tilde{K}$ has identical distribution with $\mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^{L})$. Therefore, $\mathcal{D}$'s view in Game 6 and $\mathcal{D}$'s view in Game 5 are indistinguishable with high probability.

Game 7. This game is identical to Game 6, except for responding to H_0 queries. The simulator responds to a query (K_A, K_B) on H_0 as in the box of Simulator $\mathcal{S}_7$.

The only difference between Game 6 and Game 7 occurs in the Case 4. In Game 6, the simulator responds to a random H_0 query (K_A, K_B) with $H_0(K_A, K_B)$; while in Game 7, the simulator responds with a random string K in $\{0,1\}^{n_3(\lambda)}$. Due to definition, the only case that the adversary $\mathcal{D}$ queries H_0 with (K_A, K_B) is when $\mathcal{D}$ knows nothing of $H_0(K_A, K_B)$. Therefore, from the adversary $\mathcal{D}$'s view, $H_0(K_A, K_B)$ is uniformly distributed in $\{0,1\}^{n_3(\lambda)}$. Note that K is also uniformly distributed in $\{0,1\}^{n_3(\lambda)}$, which implies that with high probability $\mathcal{D}$'s view in Game 7 and $\mathcal{D}$'s view in Game 6 are indistinguishable.

Simulator $\mathcal{S}_6$

$\underline{\mathcal{S}_6^{\mathsf{EOT_{B_2}}}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)\text{:}}$

Case 1: if $\exists(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}}$,
  return $K_B, \tilde{K_B}$;
Case 2: if $\exists(\mathrm{SK}, \overline{\mathrm{SK}}) \in T_{H_1} \wedge (K_A, K_B, K) \in T_{H_0} \wedge (\mathrm{SK}, 0, \widetilde{\mathrm{PK}}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$, s.t. $\mathit{\Pi}.\mathsf{symSHK}(\mathrm{SK}, \widetilde{\mathrm{PK}}^L \| \widetilde{\mathrm{PK}}^R) = K$,
  sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{B_2}} = T_{\mathrm{OT}_{B_2}} \cup (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B})$;
  return $K_B, \tilde{K_B}$;
Case 3: if $\exists(\mathrm{SK}', 0, \widetilde{\mathrm{PK}}^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}', 0, \mathrm{PK}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$,
  sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{B_2}} = T_{\mathrm{OT}_{B_2}} \cup (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_A, \tilde{K_B})$;
  return $K_A, \tilde{K_B}$;
Case 4: if $\exists(\mathrm{SK}', 1, \widetilde{\mathrm{PK}}^L) \in T_{\mathrm{OT}_{A_1}} \wedge (\overline{\mathrm{SK}}, \mathrm{PK}^R) \in T_{\mathrm{OT}_{B_1}} \wedge (\mathrm{SK}', 1, \mathrm{PK}^R, K_A) \in T_{\mathrm{OT}_{A_2}}$,
  sample $\tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{B_2}} = T_{\mathrm{OT}_{B_2}} \cup (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, \tilde{K_B}, K_A)$;
  return $\tilde{K_B}, K_A$;
Case 5: otherwise, randomly sample $K_B, \tilde{K_B} \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{\mathrm{OT}_{B_2}} = T_{\mathrm{OT}_{B_2}} \cup (\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L, K_B, \tilde{K_B})$;
  return $K_B, \tilde{K_B}$.

Simulator $\mathcal{S}_7$

$\underline{\mathcal{S}_7^{H_0}(K_A, K_B)\text{:}}$

Case 1: if $\exists(K_A, K_B, K) \in T_{H_0}$,
  return K;
Case 2: if $\exists(\mathrm{SK}_0, 0, \mathrm{PK}_1^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_0, \mathrm{PK}_1^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_0, \overline{\mathrm{SK}}_0) \in T_{H_1}$,
  query $\mathit{\Pi}.\mathsf{symSHK}$ with $\mathrm{SK}_0, \mathrm{PK}_1^L \| \mathrm{PK}_1^R$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
  return K;
Case 3: if $\exists(\mathrm{SK}_1, 0, \mathrm{PK}_0^R, K_A) \in T_{\mathrm{OT}_{A_2}} \wedge (\overline{\mathrm{SK}}_1, \mathrm{PK}_0^L, K_B, \tilde{K_B}) \in T_{\mathrm{OT}_{B_2}} \wedge (\mathrm{SK}_1, \overline{\mathrm{SK}}_1) \in T_{H_1}$,
  query $\mathit{\Pi}.\mathsf{symSHK}$ with $\mathrm{SK}_1, \mathrm{PK}_0^L \| \mathrm{PK}_0^R$, obtain K; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
  return K;
Case 4: otherwise, $K \leftarrow \{0,1\}^{n_3(\lambda)}$; $T_{H_0} = T_{H_0} \cup (K_A, K_B, K)$;
  return K.

Game 8. In Game 7, the queries to the adversarial interfaces are answered by the tables which are maintained by the simulator and by making queries to $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$. The simulator never make queries directly to $H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2}$; these oracles are *only* used to answer the $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$ queries (either generated by the adversary or by the simulator's response to $H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2}$ queries). At this point, we can replace the calls to $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$ with the calls to ideal algorithms $\mathsf{I.symKG}, \mathsf{I.symSHK}$ respectively, resulting in Game 8.

We note that in Game 7, the simulator is efficient, and it responds to the adversarial interfaces just by keeping several tables and calling $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$ at the honest interfaces. Thus, we can build a simulator that responds to the honest and adversarial queries precisely as the simulator does in Game 7. The result is that the view in Game 8 is identical to the view in the ideal world, and it suffices to prove that any adjacent games are indistinguishable. Next we give the rigorous proof for the indistinguishably between each adjacent games.

Simulator In Ideal Game. Let $(\mathsf{I.symKG}, \mathsf{I.symSHK})$ be the function pair that samples from the ideal symmetric NIKE family $\mathcal{T}_{\mathsf{symNIKE}}$, the simulator works as follows. In Game 8, the simulator in the ideal game maintains six tables in the same way as in Game 7 except that those table items that are set as the responses of $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$ are replaced by the responses of $\mathsf{I.symKG}, \mathsf{I.symSHK}$ respectively.

By definition, the simulator $\mathcal{S}$ now has access to $\mathsf{I.symKG}, \mathsf{I.symSHK}$ at the honest interfaces. And for the adversarial queries, $\mathcal{S}$ responds the same way as in Game 7, by just using the tables and querying the honest interfaces.

Now we prove the indistinguishability between any adjacent games.

Claim 1. Game Real $\approx$ Game 0.

Proof. Recalling that the only difference between Game Real and Game 0 is that, in Game 0 the simulator additionally maintains several tables that are completely hidden from the adversary, hence we have

$$\Pr[\text{Game Real} = 1] = \Pr[\text{Game } 0 = 1]$$

Claim 2. Game 0 $\approx$ Game 1.

Proof. Compared to Game 0, in Game 1 the simulator maintains longer tables responds to part of the queries at the adversarial interfaces by using those tables and calling the honest interfaces; besides, in Game 1 the simulator's responses to some $\mathsf{I.EOT}_{\mathsf{B}_2}$ queries are slightly different while distributed identically with that in Game 0, without influencing the simulation of all other oracle queries. For the queries to the honest interfaces, the simulator responds by forwarding the calls and responses of the algorithms $\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK}$. Moreover, the items stored in those tables are always consistent with the real oracles $H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}$ at adversarial interfaces, and the items in the $T_{\mathrm{OT}_{\mathrm{B}_2}}$ table are indistinguishable from the responses of the $\mathsf{I.EOT}_{\mathsf{B}_1}$ oracle. Hence, the response of adversarial queries by either the real game oracles

$H_0, H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2}$ (in Game 0) or by real game oracles plus honest interfaces and tables (in Game 1) are indistinguishable, which implies

$$|\Pr[\text{Game } 0 = 1] - \Pr[\text{Game } 1 = 1]| \leq \text{negl}(\lambda)$$

Claim 3. Game 1 $\approx$ Game 2.

We note that, in order not to be distinguished by the adversary, the simulator's responses at adversarial interfaces should satisfy the consistency conditions below:

1. There exists no two SK, SK$'$ such that SK $\neq$ SK$'$ and $\mathcal{S}^{H_1}(\text{SK}) = \mathcal{S}^{H_1}(\text{SK}')$;
2. There exists no two pair (K_A, K_B), (K_0', K_1') such that $(K_A, K_B) \neq (K_0', K_1')$ and $\mathcal{S}^{H_0}(K_A, K_B) = \mathcal{S}^{H_0}(K_0', K_1')$;
3. There exists no two SK, SK$'$ (SK $\neq$ SK$'$) such that $\mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}, 0) = \mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}', 0)$ or $\mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}, 1) = \mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}', 1)$;
4. There exists no two $\overline{\text{SK}}, \overline{\text{SK}}'$ ($\overline{\text{SK}} \neq \overline{\text{SK}}'$) such that $\mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\text{SK}}) = \mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\text{SK}}')$;
5. For any SK $\in \ell_1(\lambda)$, $\Pi.\mathsf{symKG}(\text{SK}) = \mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}, 0) \| \mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_1}}(\mathcal{S}^{H_1}(\text{SK}))$;
6. For any $\text{SK}_0 \in \ell_1(\lambda)$ and $\text{PK}_1 \in \ell_2(\lambda)$, $\Pi.\mathsf{symSHK}(\text{SK}_0, \text{PK}_1) = \mathcal{S}^{H_0}(\mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_2}}(\text{SK}_0, \text{PK}_1^R, 0), \mathsf{LoR}_0(\mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_2}}(\mathcal{S}^{H_1}(\text{SK}_0), \text{PK}_1^L)))$, where $\text{PK}_1 = \text{PK}_1^L \| \text{PK}_1^R$;
7. For any $\text{SK}_0, \text{SK}_1 \in \ell_1(\lambda)$, $\mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_2}}(\text{SK}_0, \mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_1}}(\mathcal{S}^{H_1}(\text{SK}_1)), 0) = \mathsf{LoR}_0(\mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_2}}(\mathcal{S}^{H_1}(\text{SK}_1), \mathcal{S}^{\mathsf{EOT}_{\mathsf{A}_1}}(\text{SK}_0, 0)))$.

Proof. The only difference between Game 1 and Game 2 occurs in the Case 5. In Game 1, the simulator responds to a random H_1 query SK with $H_1(\text{SK})$ while in Game 2, the simulator responds with a random string $\overline{\text{SK}} \leftarrow \{0,1\}^{\ell_3(\lambda)}$ and implicitly set $\mathcal{S}^{\mathsf{EOT}_{\mathsf{B}_1}}(\overline{\text{SK}}) = \mathsf{Trun}_{\ell_4}(\Pi.\mathsf{symKG}(\text{SK}))$, where Trun_{ℓ_4} is a function that truncates the last $\ell_4(\lambda)$ bits of an input as the output.
By definition, H_1 is a random oracle, the distribution of $H_1(\text{SK})$ should be uniformly random in $\{0,1\}^{\ell_3(\lambda)}$, and $\overline{\text{SK}}$ is well-distributed. Besides, all the consistency conditions hold in either games. Therefore, from the adversary's view, with high probability Game 1 and Game 2 are indistinguishable except when $\mathcal{S}^{H_1}$ has collisions, which occurs with probability bounded by $\frac{q^2}{2^{\ell_3(\lambda)}}$. Hence, we have

$$|\Pr[\text{Game } 1 = 1] - \Pr[\text{Game } 2 = 1]| \leq \frac{q^2}{2^{\ell_3(\lambda)}} \leq \mathsf{negl}(\lambda)$$

Claim 4. Game 2 $\approx$ Game 3.

Proof. Recalling that the only difference between Game 2 and Game 3 occurs in the Case 4 and Case 5 of simulating $\mathsf{I.EOT}_{\mathsf{A}_1}$ where (SK, b) never appears in the previous queries.
For the Case 4, in Game 2, the simulator responds to a random query $(\text{SK}, 0)$ to $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $\mathsf{I.EOT}_{\mathsf{A}_1}(\text{SK}, 0)$; while in Game 3, the simulator replaces it

with PK^L being the first $2\ell_5(\lambda)$ bits of $\mathit{\Pi}.\mathsf{symKG}(\mathrm{SK})$. Due to definition, $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 0)$ and PK^L are identical. For the Case 5, in Game 2, the simulator responds to a random query $(\mathrm{SK}, 1)$ to $\mathsf{I.EOT}_{\mathsf{A}_1}$ with $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$; while in Game 3, the simulator replaces it with a random string $\tilde{\mathrm{PK}}^L$ in $\{0,1\}^{2\ell_5(\lambda)}$.By definition, the only case that the adversary queries $(\mathrm{SK}, 1)$ to $\mathsf{I.EOT}_{\mathsf{A}_1}$ is when the adversary knows nothing of $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$. From the adversary's view, $\mathsf{I.EOT}_{\mathsf{A}_1}(\mathrm{SK}, 1)$ is uniformly distributed in $\{0,1\}^{2\ell_5(\lambda)}$, thus $\tilde{\mathrm{PK}}^L$ is well-distributed. Besides, in either games, the consistency conditions hold.

Therefore, from the adversary's view, Game 2 and Game 3 are indistinguishable except when $\tilde{\mathrm{PK}}^L$ already appears in a previous entry, which occurs with negligible probability bounded by $\frac{q^2}{|2^{2\ell_5(\lambda)}|}$. Hence,

$$|\Pr[\text{Game } 2 = 1] - \Pr[\text{Game } 3 = 1]| \leq \frac{q^2}{2^{\ell_2(\lambda)}} \leq \mathsf{negl}(\lambda)$$

Claim 5. Game 3 $\approx$ Game 4.

Proof. Recalling that the only difference between Game 3 and Game 4 occurs in the Case 5 of simulating $\mathsf{I.EOT}_{\mathsf{B}_1}$. In Game 3, the simulator responds to a query $\overline{\mathrm{SK}}$ to $\mathsf{I.EOT}_{\mathsf{B}_1}$ with $\mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$; while in Game 4, the simulator replaces it with PK^R which is the last $\ell_4(\lambda)$ bits of $\mathit{\Pi}.\mathsf{symKG}(\mathrm{SK})$ for a randomly sampled string SK in $\{0,1\}^{\ell_1(\lambda)}$.Due to definition, $\mathsf{I.EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})$ and PK^R have identical distribution. Besides, for $H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_2}, H_0$ queries, the simulator's responses are identical in either game, both satisfying the consistency conditions. Therefore, from the adversary's view, Game 3 and Game 4 are indistinguishable.

Claim 6. Game 4 $\approx$ Game 5.

Proof. Recalling that the only difference between Game 4 and Game 5 occurs in the Case 3 and Case 4 of simulating the responses of $\mathsf{I.EOT}_{\mathsf{A}_2}$.

For the Case 3, in Game 4, the simulator responds to a random query $\mathrm{SK}_0, 0, \mathrm{PK}_1^R$ with $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, 0, \mathrm{PK}_1^R)$; while in Game 5, the simulator replaces it with a random string K_A in $\mathsf{DomSym}_{\mathrm{K}}$. By definition, the distributions of $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, 0, \mathrm{PK}_1^R)$ and K_A are identical. For the Case 4, in Game 4, the simulator responds to a random query $\mathrm{SK}_0, 1, \mathrm{PK}_1^R$ with $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, 1, \mathrm{PK}_1^R)$; while in Game 5, the simulator replaces it with a random string $\tilde{K}$ in $\mathsf{DomSym}_{\mathrm{K}}$. By definition, the distributions of $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, 1, \mathrm{PK}_1^R)$ and $\tilde{K}$ are identical.

Besides, for $H_1, \mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}, \mathsf{I.EOT}_{\mathsf{B}_2}, H_0$ queries, the simulator's responses are identical in either game, both satisfying the consistency conditions. Therefore, from the adversary's view, Game 4 and Game 5 are indistinguishable. We have

$$|\Pr[\text{Game } 4 = 1] - \Pr[\text{Game } 5 = 1]| \leq \frac{q^2}{2^{n_3(\lambda)}} \leq \mathsf{negl}(\lambda)$$

Claim 7. Game 5 $\approx$ Game 6.

Proof. Recalling that the only difference between Game 5 and Game 6 occurs in the Case 5 of simulating the responses of $\mathsf{I.EOT_{B_2}}$.In Game 5, the simulator responds to a $\mathsf{I.EOT_{B_2}}$ query $(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)$ with $\mathsf{I.EOT_{B_2}}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)$; while in Game 6, the simulator replaces it with a random string $K_B \| \tilde{K_B}$ from $\{0,1\}^{n_3(\lambda)}$. Due to definition, the distributions of $\mathsf{I.EOT_{B_2}}(\overline{\mathrm{SK}}, \widetilde{\mathrm{PK}}^L)$ and the random string $K_B \| \tilde{K_B}$ are identical. And for $H_1, \mathsf{I.EOT_{A_1}}, \mathsf{I.EOT_{B_1}}, \mathsf{I.EOT_{A_2}}, H_0$ queries, the responses are identical in either game, both satisfying the consistency conditions. Therefore, from the adversary's view, Game 5 and Game 6 are indistinguishable with high probability, except when a collision for $\mathsf{I.EOT_{B_2}}$ occurs, hence we have

$$|\Pr[\text{Game } 5 = 1] - \Pr[\text{Game } 6 = 1]| \leq \frac{q^2}{2^{n_3(\lambda)}} \leq \mathsf{negl}(\lambda)$$

Claim 8. Game 6 $\approx$ Game 7.

Proof. Recalling that the only difference between Game 6 and Game 7 occurs in the Case 4 of simulating H_0. In Game 6, the simulator responds to a random query (K_A, K_B) with $H_0(K_A, K_B)$; while in Game 7, the simulator replaces it with a random string K uniformly distributed in $\{0,1\}^{n_3(\lambda)}$.
Due to definition, the only case that the adversary queries (K_A, K_B) to H_0 is when the adversary knows nothing of $H_0(K_A, K_B)$, thus from the adversary's view, $H_0(K_A, K_B)$ is uniformly distributed in $\{0,1\}^{n_3(\lambda)}$, and K is well-distributed. And for $H_1, \mathsf{I.EOT_{A_1}}, \mathsf{I.EOT_{B_1}}, \mathsf{I.EOT_{A_2}}, \mathsf{I.EOT_{B_2}}$ queries, the responses are identical in either game, both satisfying the consistency conditions.
Therefore, from the adversary's view, Game 6 and Game 7 are indistinguishable unless collision event for H_0 occurs, which is bounded by $\frac{q^2}{|\{0,1\}^{n_3(\lambda)}|}$. Hence we have

$$|\Pr[\text{Game } 6 = 1] - \Pr[\text{Game } 7 = 1]| \leq \frac{q^2}{2^{n_3(\lambda)}} \leq \mathsf{negl}(\lambda)$$

Claim 9. Game 7 $\approx$ Game 8.

Proof. Let $(\mathsf{I.symKG}, \mathsf{I.symSHK})$ be the function pair that samples from $\mathcal{T}_{\mathsf{symNIKE}}$. We note that in Game 7, the simulator responds all of the adversarial interfaces just using tables and the algorithms $(\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$ at honest interfaces, it never directly calls the real oracles at honest interfaces. We immediately observe that $\mathcal{S}_8$ is identical to the simulator in the ideal game, which refers to

$$|\Pr[\text{Game } 8 = 1] - \Pr[\text{Ideal Game} = 1]|$$

Therefore, it is rest to prove that Game 7 and Game 8 are close.

H_0, H_1 are random oracles, $\mathsf{I.EOT}_{\mathsf{A}_1}, \mathsf{I.EOT}_{\mathsf{B}_1}$ are random injections, $\mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_2}$ are random algorithms satisfying $\mathsf{I.EOT}_{\mathsf{A}_2}(\mathrm{SK}_0, b, \mathsf{EOT}_{\mathsf{B}_1}(\overline{\mathrm{SK}})) = \mathsf{LoR}_b(\mathsf{I.EOT}_{\mathsf{B}_2}(\overline{\mathrm{SK}}, \mathsf{EOT}_{\mathsf{A}_1}(\mathrm{SK}_0, b)))$ for any SK_0 in $\{0,1\}^{\ell_1(\lambda)}$, $\overline{\mathrm{SK}}$ in $\{0,1\}^{\ell_3(\lambda)}$ and $b \in \{0,1\}$.

Conditioned on the oracles H_1, H_0 have no collisions, for any $\mathrm{SK}_0, \mathrm{SK}_1$ in $\{0,1\}^{\ell_1(\lambda)}$, it's oblivious that the distributions of $\Pi.\mathsf{symKG}(\mathrm{SK}_0)$ and $\mathsf{I.symKG}(\mathrm{SK}_0)$ are identical, and the distributions of $\Pi.\mathsf{symSHK}(\mathrm{SK}_1, \Pi.\mathsf{symKG}(\mathrm{SK}_0))$ and $\mathsf{I.symSHK}(\mathrm{SK}_1, \mathsf{I.symKG}(\mathrm{SK}_0))$ are identical; and the shared key property holds for both $\Pi.\mathsf{symSHK}$ and $\mathsf{I.symSHK}$.

Let $\mathsf{Collision}$ denote the event that there exists collision in the oracles $H_1, H_0, \mathsf{I.EOT}_{\mathsf{A}_2}, \mathsf{I.EOT}_{\mathsf{B}_2}$. The probability that H_1 has collision is bounded by $\frac{q^2}{2^{\ell_3(\lambda)}}$; The probability that H_0 has collision is bounded by $\frac{q^2}{2^{n_3(\lambda)}}$; The probability that $\mathsf{I.EOT}_{\mathsf{A}_2}$ has collision is bounded by $\frac{q^2}{2^{n_3(\lambda)}}$; The probability that $\mathsf{I.EOT}_{\mathsf{B}_2}$ has collision is bounded by $\frac{q^2}{2^{2n_3(\lambda)}}$.

If none of the collision occurs, we can replace $(\Pi.\mathsf{symKG}, \Pi.\mathsf{symSHK})$ with $(\mathsf{I.symKG}, \mathsf{I.symSHK})$, which represents Game 8. Moreover, we can bound the probability of collision by

$$\Pr[\mathsf{Collision}] \leq \frac{q^2}{2^{\ell_3(\lambda)}} + \frac{q^2}{2^{n_3(\lambda)}} + \frac{q^2}{2^{n_3(\lambda)}} + \frac{q^2}{2^{2n_3(\lambda)}} \leq \mathsf{negl}(\lambda),$$

which refers to

$$|\Pr[\text{Game } 7 = 1] - \Pr[\text{Game } 8 = 1]| \leq \Pr[\mathsf{Collision}] \leq \mathsf{negl}(\lambda)$$

Combining all claims together, we have

$$|\Pr[\text{Real Game} = 1] - \Pr[\text{Ideal Game} = 1]| \leq \frac{q^2}{2^{\ell_2(\lambda)}} + \frac{2q^2}{2^{\ell_3(\lambda)}} + \frac{5q^2}{2^{n_3(\lambda)}} + \frac{q^2}{2^{2n_3(\lambda)}} \leq \mathsf{negl}(\lambda),$$

thus we complete the entire proof.

D Proof of Theorem 4 and Theorem 5

D.1 Proof of Theorem 4

Proof. In the proof, the simulator can directly use information from the queries to adversarial interface as the queries for honest interface, such that the simulator's responses to adversarial interface is consistent with that of honest interfaces. Therefore, no PPT algorithm can distinguish real world game and ideal world game, namely.

The simulator $\mathcal{S}$ in the ideal game has the external oracle access to ideal PKE scheme $\mathsf{I.PKE} = (\mathsf{I.KGen}, \mathsf{I.Enc}, \mathsf{I.Dec})$; the simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$.

Simulator $\mathcal{S}$

$\mathcal{S}^{\mathsf{OT}_1}(\mathrm{SK}, b)$:

Case 1: if $\exists(\mathrm{SK}, b, \mathrm{PK}) \in T_{\mathrm{OT}_1}$,
return PK;
Case 2: if $b = 0$, query $\mathsf{I.KGen}$ with SK and obtain PK;
return PK;
Case 3: otherwise, $\widetilde{\mathrm{PK}} \leftarrow \mathsf{Y}$; $T_{\mathrm{OT}_1} = T_{\mathrm{OT}_1} \cup (\mathrm{SK}, b, \widetilde{\mathrm{PK}})$,
return $\widetilde{\mathrm{PK}}$.

$\mathcal{S}^{\mathsf{OT}_2}(m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}})$:

Case 1: if $\exists(m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}}, C) \in T_{\mathrm{OT}_2}$,
return C;
Case 2: if $m_0 = m_1$, query $\mathsf{I.Enc}$ with $(\mathrm{PK}, m_0, \overline{\mathrm{SK}})$ and obtain C; $T_{\mathrm{OT}_2} = T_{\mathrm{OT}_2} \cup (m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}}, C)$,
return C;
Case 3: if $m_0 \neq m_1$, randomly sample $C \leftarrow \mathcal{C}$, $T_{\mathrm{OT}_2} = T_{\mathrm{OT}_2} \cup (m_0, m_1, \mathrm{PK}, \overline{\mathrm{SK}}, C)$,
return C.

$\mathcal{S}^{\mathsf{OT}_3}(\mathrm{SK}, b, C)$:

Case 1: if $\exists(\mathrm{SK}, b, C, m) \in T_{\mathrm{OT}_3}$,
return m;
Case 2: if $b = 0$, query $\mathsf{I.Dec}$ with (SK, C) and obtain m; $T_{\mathrm{OT}_3} = T_{\mathrm{OT}_3} \cup (\mathrm{SK}, b, C, m)$,
return m;
Case 3: if $b = 1$, randomly sample $\tilde{m} \leftarrow \mathcal{M}$, $T_{\mathrm{OT}_3} = T_{\mathrm{OT}_3} \cup (\mathrm{SK}, b, C, \tilde{m})$,
return $\tilde{m}$.

D.2 Proof of Theorem 5

Proof. The simulator in the ideal game is described below. The simulator $\mathcal{S}$ has the external oracle access to ideal OT protocol $\mathsf{I.2OT} = (\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3)$; the simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$.

Simulator $\mathcal{S}$

$\underline{\mathcal{S}^{H_0}(\overline{\mathrm{SK}}, m_1)\text{:}}$

Case 1: if $\exists(\overline{\mathrm{SK}}, m_1, r_0) \in T_{H_0}$,
    return r_0;
Case 2: if $\exists(\mathrm{PK}_0, m_0, r_0, C_0) \in T_{\mathrm{Enc}} \wedge (\mathrm{PK}_1, m_1, r_1, C_1) \in T_{\mathrm{Enc}} \wedge (K_0, \mathrm{PK}_0, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (K_1, \mathrm{PK}_1, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$, s.t. $\mathsf{I.OT}_2(Q_0 \| Q_1, m_0, m_1, \overline{\mathrm{SK}}) = w$,
    return r_0;

Case 3: otherwise, randomly sample $r_0 \leftarrow \mathcal{R}$; $T_{H_0} = T_{H_0} \cup (\overline{\text{SK}}, m_1, r_0)$;
 return r_0.

$\underline{\mathcal{S}^{H_1}(\overline{\text{SK}}, m_0):}$

Case 1: if $\exists(\overline{\text{SK}}, m_0, r_1) \in T_{H_1}$,
 return r_1;
Case 2: if $\exists(\text{PK}_0, m_0, r_0, C_0) \in T_{\text{Enc}} \wedge (\text{PK}_1, m_1, r_1, C_1) \in T_{\text{Enc}} \wedge (K_0, \text{PK}_0, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (K_1, \text{PK}_1, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$, s.t. $\mathsf{I.OT}_2(Q_0 \| Q_1, m_0, m_1, \overline{\text{SK}}) = w$,
 return r_1;
Case 3: otherwise, randomly sample $r_1 \leftarrow \mathcal{R}$; $T_{H_1} = T_{H_1} \cup (\overline{\text{SK}}, m_0, r_1)$;
 return r_1.

$\underline{\mathcal{S}^{H_2}(\text{SK}, b):}$

Case 1: if $\exists(\text{SK}, b, Q_{1-b}) \in T_{H_2}$,
 return Q_{1-b};
Case 2: if $\exists(Q_{1-b}, K) \in T_{H_3} \wedge (K, \text{PK}, Q_{1-b}) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (\text{SK}, \text{PK}) \in T_{\text{KGen}}$,
 return Q_{1-b};
Case 3: if $\exists(Q_{1-b}, K) \in T_{H_3} \wedge (K, \text{PK}, Q_{1-b}) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (\text{PK}, m, r, C) \in T_{\text{Enc}} \wedge (\text{SK}, C, m) \in T_{\text{Dec}}$,
 return Q_{1-b};
Case 4: otherwise, query $\mathsf{I.OT}_1$ with (SK, b), obtain $Q = Q_0 \| Q_1$; $T_{H_2} = T_{H_2} \cup (\text{SK}, b, Q_{1-b})$;
 return Q_{1-b}.

$\underline{\mathcal{S}^{H_3}(Q):}$

Case 1: if $\exists(Q, K) \in T_{H_3}$,
 return K;
Case 2: if $\exists(K, \text{PK}, Q) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
return K;
Case 3: otherwise, randomly sample $K \leftarrow \mathcal{K}$, $T_{H_3} = T_{H_3} \cup (Q, K)$.

$\underline{\mathcal{S}^{\mathcal{E}}(K, \text{PK}):}$

Case 1: if $\exists(K, \text{PK}, Q) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
 return Q;
Case 2: if $\exists(\text{SK}, b, Q_{1-b}) \in T_{H_2} \wedge (Q_{1-b}, K) \in T_{H_3} \wedge (\text{SK}, \text{PK}) \in T_{\text{KGen}}$,
query $\mathsf{I.OT}_1$ with (SK, b), obtain $Q = Q_0 \| Q_1$, $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (K, \text{PK}, Q_b)$,
return Q_b;
Case 3: otherwise, randomly sample $Q \leftarrow \mathcal{C}$, $T_{\mathcal{E}} = T_{\mathcal{E}} \cup (K, \text{PK}, Q)$,
return Q.

$\underline{\mathcal{S}^{\mathcal{E}^{-1}}(K, Q):}$

Case 1: if $\exists(K, \text{PK}, Q) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
 return PK;
Case 2: if $\exists(\text{SK}, b, Q_{1-b}) \in T_{H_2} \wedge (Q_{1-b}, K) \in T_{H_3}$, s.t. $\mathsf{I.OT}_1(\text{SK}, b) = (Q, Q_1 - b)$ or $\mathsf{I.OT}_1(\text{SK}, b) = (Q_1 - b, Q)$,
run $\mathcal{S}^{\mathsf{KGen}}(\text{SK})$ and obtain PK,
 return PK;

Case 3: otherwise, randomly sample $\mathrm{PK} \leftarrow \mathcal{PK}$, $T_{\mathcal{E}^{-1}} = T_{\mathcal{E}^{-1}} \cup (K, \mathrm{PK}, Q)$, return Q.

$\underline{\mathcal{S}^{P_0}(C_0 \| C_1)}$:

Case 1: if $\exists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
 return w;
Case 2: if $\exists (\mathrm{PK}_0, m_0, r_0, C_0) \in T_{\mathrm{Enc}} \wedge (\mathrm{PK}_1, m_1, r_1, C_1) \in T_{\mathrm{Enc}} \wedge (K_0, \mathrm{PK}_0, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (K_1, \mathrm{PK}_1, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (\overline{\mathrm{SK}}, m_1, r_0) \in T_{H_0}) \wedge (\overline{\mathrm{SK}}, m_0, r_1) \in T_{H_1})$, query $\mathsf{I.OT}_2$ with $(Q_0 \| Q_1, m_0, m_1, \overline{\mathrm{SK}})$, obtain w; return w;
Case 3: otherwise, randomly sample $w \leftarrow \mathcal{C} \times \mathcal{C}$, $T_{P_0} = T_{P_0} \cup (C_0, C_1, w)$, return w.

$\underline{\mathcal{S}^{P^{-1}}(w)}$:

Case 1: if $\exists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
 return C_0, C_1;
Case 2: if $\exists (\mathrm{PK}_0, m_0, r_0, C_0) \in T_{\mathrm{Enc}} \wedge (\mathrm{PK}_1, m_1, r_1, C_1) \in T_{\mathrm{Enc}} \wedge (K_0, \mathrm{PK}_0, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (K_1, \mathrm{PK}_1, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge (\overline{\mathrm{SK}}, m_1, r_0) \in T_{H_0}) \wedge (\overline{\mathrm{SK}}, m_0, r_1) \in T_{H_1})$, s.t.$\mathsf{I.OT}_2(Q_0 \| Q_1, m_0, m_1, \overline{\mathrm{SK}}) = w$,
 return $C_0 \| C_1$;
Case 3: otherwise, randomly sample $C_0, C_1 \leftarrow \mathcal{C}$, $T_{P_0^{-1}} = T_{P_0^{-1}} \cup (C_0, C_1, w)$, return C_0, C_1.

$\underline{\mathcal{S}^{\mathsf{KGen}}(\mathrm{SK})}$:

Case 1: if $\exists (\mathrm{SK}, \mathrm{PK}) \in T_{\mathrm{KGen}}$,
 return PK;
Case 2: if $\exists (\mathrm{PK}, m, r, C) \in T_{\mathrm{Enc}}$ and $\exists (\mathrm{SK}, C, m) \in T_{\mathrm{Dec}}$,
 return PK;
Case 3: if $\exists (\mathrm{SK}, b, Q_{1-b}) \in T_{H_2} \wedge \exists (Q_{1-b}, K) \in T_{H_3} \wedge \exists (K, PK, Q_{1-b}) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,
 return PK;
Case 4: otherwise, sample $\mathrm{PK} \leftarrow \mathcal{PK}$, $T_{\mathrm{KGen}} = T_{\mathrm{KGen}} \cup (\mathrm{SK}, \mathrm{PK})$;
 return PK.

$\underline{\mathcal{S}^{\mathsf{Enc}}(\mathrm{PK}, m, r)}$:

Case 1: if $\exists (\mathrm{PK}, m, r, C) \in T_{\mathrm{Enc}}$,
 return C;
Case 2: if $\exists (\mathrm{SK}, \mathrm{PK}) \in T_{\mathrm{KGen}}$ and $\exists (\mathrm{SK}, C, m) \in T_{\mathrm{Dec}}$,
 return C;
Case 3: if $\exists (\overline{\mathrm{SK}}, m_1, r_0) \in T_{H_0} \wedge (K, PK, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge \exists (\tilde{K}, \tilde{PK}, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$
Subcase 3.1: if $\exists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$, , s.t. $\mathsf{I.OT}_2(Q_0 \| Q_1, m, m_1, \overline{\mathrm{SK}}) = w$; $T_{\mathrm{Enc}} = T_{\mathrm{Enc}} \cup (\mathrm{PK}, m, r, C_0)$;
 return C_0;
Subcase 3.2: if $\nexists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
quey $\mathsf{I.OT}_2$ with $(Q_0 \| Q_1, m, m_1, \overline{\mathrm{SK}})$ and obtain w, run $\mathcal{S}^{P_0^{-1}(w)}$ and obtain $W = (C_0, C_1)$, $T_{P_0^{-1}} = T_{P_0^{-1}} \cup (C_0, C_1, w)$, $T_{\mathrm{Enc}} = T_{\mathrm{Enc}} \cup (\mathrm{PK}, m, r, C_0)$;
 return C_0;
Case 4: if if $\exists (\overline{\mathrm{SK}}, m_0, r_1) \in T_{H_1} \wedge (K, PK, Q_0) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}} \wedge \exists (\tilde{K}, \tilde{PK}, Q_1) \in T_{\mathcal{E}} \cup T_{\mathcal{E}^{-1}}$,

Subcase 4.1: if $\exists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$ s.t. $\mathsf{I.OT}_2(Q_0 \| Q_1, m_0, m, \overline{\mathrm{SK}}) = w$,
$T_{\mathrm{Enc}} = T_{\mathrm{Enc}} \cup (\mathrm{PK}, m, r, C_1)$;
return C_1;
Subcase 4.2: if $\nexists (C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
quey $\mathsf{I.OT}_2$ with $(Q_0 \| Q_1, m_0, m, \overline{\mathrm{SK}})$ and obtain w, run $\mathcal{S}^{P_0^{-1}(w)}$ and obtain $W = (C_0, C_1)$,
$T_{P_0^{-1}} == T_{P_0^{-1}} \cup (C_0, C_1, w)$, $T_{\mathrm{Enc}} = T_{\mathrm{Enc}} \cup (\mathrm{PK}, m, r, C_1)$;
return C_1;
Case 5: otherwise, randomly sample $C \leftarrow \mathcal{C}$, $T_{\mathrm{Enc}} = T_{\mathrm{Enc}} \cup (\mathrm{PK}, m, r, C)$;
return C;

$\underline{\mathcal{S}^{\mathsf{Dec}}(\mathrm{SK}, C)}$:
Case 1: if $\exists (\mathrm{SK}, C, m) \in T_{\mathrm{Dec}}$,
return m;
Case 2: if $\exists (\mathrm{PK}, m, r, C) \in T_{\mathrm{Enc}}$ and $\exists (\mathrm{SK}, \mathrm{PK}) \in T_{\mathrm{KGen}}$,
return m;
Case 3: if $\exists (C, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
query $\mathsf{I.OT}_3$ with $(w, \mathrm{SK}, 0)$ and obtain m_0; $T_{\mathrm{Dec}} = T_{\mathrm{Dec}} \cup (\mathrm{SK}, C, m_0)$;
return m_0;
Case 4: if $\exists (C_0, C, w) \in T_{P_0} \cup T_{P_0^{-1}}$,
query $\mathsf{I.OT}_3$ with $(w, \mathrm{SK}, 1)$ and obtain m_1; $T_{\mathrm{Dec}} = T_{\mathrm{Dec}} \cup (\mathrm{SK}, C, m_1)$;
return m_1;
Case 5: otherwise, randomly sample $m \leftarrow \mathcal{M}$, $T_{\mathrm{Dec}} = T_{\mathrm{Dec}} \cup (\mathrm{SK}, C, m)$,
return m.

E Proof of Theorem 6

Proof. Here, we give full proof of Theorem 6, that the constructed $\Pi.\mathsf{2OT}$ in Sec. 5 is indifferentiable from ideal two round OT.

In the real world, the differentiator $\mathcal{D}$ has oracle access to $(\Pi.\mathsf{OT}_1, \Pi.\mathsf{OT}_2, \Pi.\mathsf{OT}_3)$ via the honest interface and oracle access to (H_0, $\mathcal{E}_1$, $\mathcal{E}_1^{-1}$, $\mathcal{E}_2$, $\mathcal{E}_2^{-1}$, $\mathsf{I.EOT}_{\mathsf{A}_1}$, $\mathsf{I.EOT}_{\mathsf{A}_2}$, $\mathsf{I.EOT}_{\mathsf{B}_1}$, $\mathsf{I.EOT}_{\mathsf{B}_2}$) via the adversarial interface. In contrast, in the ideal world, the differentiator $\mathcal{D}$ has oracle access to $(\mathsf{I.OT}_1, \mathsf{I.OT}_2, \mathsf{I.OT}_3)$ via the honest interface and access to $\mathcal{S}$ via the adversarial interface. Therefore, to establish a proof, we need to build an explicit (and efficient) simulator $\mathcal{S}$ that simulates the rest oracles (H_0, $\mathcal{E}_1$, $\mathcal{E}_1^{-1}$, $\mathcal{E}_2$, $\mathcal{E}_2^{-1}$, $\mathsf{I.EOT}_{\mathsf{A}_1}$, $\mathsf{I.EOT}_{\mathsf{A}_2}$, $\mathsf{I.EOT}_{\mathsf{B}_1}$, $\mathsf{I.EOT}_{\mathsf{B}_2}$) properly by making queries to ($\mathsf{I.OT}_1$, $\mathsf{I.OT}_2$, $\mathsf{I.OT}_3$).

Namely, for any PPT differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real game is computationally close to the view in the ideal game. To do so, we will go through with a sequence of hybrid games, where in each game, the simulator responds to all of the queries (both honest and adversarial) in a slightly different way and the last game is the same as the ideal world. Note that the differentiator $\mathcal{D}$ can make at most q queries to the oracles, where $q = \mathrm{poly}(\lambda)$.

First, we describe our simulator $\mathcal{S}$ in the ideal game.

Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\underline{\mathcal{S}^{H_0}(Q, m_0, m_1, \widetilde{\mathrm{SK}})}$:

Case 1: if $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$,
  return e;
Case 2: if $\exists(*, e, *, *, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$,
  return e;
Case 3: otherwise, query the external $\mathsf{I.OT}_2$ with $(Q, m_0, m_1, \widetilde{\mathrm{SK}})$ to obtain e;
  $T_{H_0} = T_{H_0} \cup (Q, m_0, m_1, \widetilde{\mathrm{SK}}, e)$;
  return e.

$\underline{\mathcal{S}^{\mathcal{E}_1}(e, \widetilde{\mathrm{PK}})}$:

Case 1: if $\exists(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}}) \in T_{\mathcal{E}_1} \cup T_{\mathcal{E}_1^{-1}}$,
  return $\widehat{\mathrm{PK}}$;
Case 2: if $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, \widetilde{\mathrm{PK}}) \in T_{\mathsf{EOT}_{\mathsf{B}_1}}$ and $\exists(\widehat{\mathrm{PK}}, e, *, *, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$,
  return $\widehat{\mathrm{PK}}$;
Case 3 : otherwise, randomly sample $\widehat{\mathrm{PK}}$, $T_{\mathcal{E}_1} = T_{\mathcal{E}_1} \cup (e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}})$;
  return $\widehat{\mathrm{PK}}$.

$\underline{\mathcal{S}^{\mathcal{E}_1^{-1}}(e, \widehat{\mathrm{PK}})}$:

Case 1: if $\exists(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}}) \in T_{\mathcal{E}_1} \cup T_{\mathcal{E}_1^{-1}}$,
  return $\widetilde{\mathrm{PK}}$;
Case 2: if $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, \widetilde{\mathrm{PK}}) \in T_{\mathsf{EOT}_{\mathsf{B}_1}}$ and $\exists(\widehat{\mathrm{PK}}, e, *, *, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$,
  return $\widetilde{\mathrm{PK}}$;
Case 3 : otherwise, randomly sample $\widetilde{\mathrm{PK}}$, $T_{\mathcal{E}_1^{-1}} = T_{\mathcal{E}_1^{-1}} \cup (e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}})$;
  return $\widetilde{\mathrm{PK}}$.

$\underline{\mathcal{S}^{\mathcal{E}_2}(K, m)}$:

Case 1: if $\exists(K, m, C) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$,
  return C;
Case 2: if $\exists(Q, m, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, Q, K, K_1) \in T_{\mathsf{EOT}_{\mathsf{B}_2}}$ and $\exists(\widehat{\mathrm{PK}}, e, C, *, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m, m_1, \widetilde{\mathrm{SK}})$,
  return C;
Case 3: if $\exists(Q, m_0, m, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, Q, K_0, K) \in T_{\mathsf{EOT}_{\mathsf{B}_2}}$ and $\exists(\widehat{\mathrm{PK}}, e, *, C, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m, \widetilde{\mathrm{SK}})$,
  return C;
Case 4: otherwise, randomly sample C; $T_{\mathcal{E}_2} = T_{\mathcal{E}_2} \cup (K, m, C)$;
  return C.

$\underline{\mathcal{S}^{\mathcal{E}_2^{-1}}(K, C)}$:

Case 1: if $\exists(K, m, C) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$,
return m;
Case 2: if $\exists(Q, m, m_1, \widetilde{\text{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\text{SK}}, Q, K, K_1) \in T_{\mathsf{EOT}_{\text{B}_2}}$ and $\exists(\widetilde{\text{PK}}, e, C, *, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m, m_1, \widetilde{\text{SK}})$,
return m;
Case 3: if $\exists(Q, m_0, m, \widetilde{\text{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\text{SK}}, Q, K_0, K) \in T_{\mathsf{EOT}_{\text{B}_2}}$ and $\exists(\widetilde{\text{PK}}, e, *, C, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m, \widetilde{\text{SK}})$,
return m;
Case 4: otherwise, randomly sample m; $T_{\mathcal{E}_2^{-1}} = T_{\mathcal{E}_2^{-1}} \cup (K, m, C)$;
return m.

$\underline{\mathcal{S}^{\text{EOT}_{\text{A}_1}}(\text{SK}, b):}$

Case 1: if $\exists(\text{SK}, b, Q) \in T_{\text{EOT}_{\text{A}_1}}$,
return Q;
Case 2: otherwise, query $\mathsf{I.OT}_1(\text{SK}, b)$, obtain Q; $T_{\text{EOT}_{\text{A}_1}} = T_{\text{EOT}_{\text{A}_1}} \cup (\text{SK}, b, Q)$;
return Q.

$\underline{\mathcal{S}^{\text{EOT}_{\text{B}_1}}(\widetilde{\text{SK}})):}$

Case 1: if $\exists(\widetilde{\text{SK}}, \widetilde{\text{PK}}) \in T_{\text{EOT}_{\text{B}_1}}$,
return $\widetilde{\text{PK}}$;
Case 2: if $\exists(\text{SK}, 0, \widetilde{\text{PK}}, K_0) \cup (\text{SK}, 1, \widetilde{\text{PK}}, K_1) \in T_{\text{EOT}_{\text{A}_2}}$ and $\exists(\widetilde{\text{SK}}, Q, K_0, K_1) \in T_{\mathsf{EOT}_{\text{B}_2}}$,
return $\widetilde{\text{PK}}$;
Case 3: otherwise, randomly sample $\widetilde{\text{PK}}$; $T_{\text{EOT}_{\text{B}_1}} = T_{\text{EOT}_{\text{B}_1}} \cup (\widetilde{\text{SK}}, \widetilde{\text{PK}})$;
return $\widetilde{\text{SK}}, \widetilde{\text{PK}}$.

$\underline{\mathcal{S}^{\text{EOT}_{\text{A}_2}}(\text{SK}, b, \widetilde{\text{PK}}):}$

Case 1: if $\exists(\text{SK}, b, \widetilde{\text{PK}}, K_b) \in T_{\text{EOT}_{\text{A}_2}}$,
return K_b;
Case 2: if $\exists(\text{SK}, b, Q) \in T_{\text{EOT}_{\text{A}_1}}$ and $(\widetilde{\text{SK}}, \widetilde{\text{PK}}) \in T_{\mathsf{EOT}_{\text{B}_1}}$ and $(\widetilde{\text{SK}}, Q, K_0, K_1) \in T_{\mathsf{EOT}_{\text{B}_2}}$,
return K_b;
Case 3: otherwise, randomly sample K_b, $T_{\text{EOT}_{\text{A}_2}} = T_{\text{EOT}_{\text{A}_2}} \cup (\text{SK}, b, \widetilde{\text{PK}}, K_b)$;
return K_b;

$\underline{\mathcal{S}^{\text{EOT}_{\text{B}_2}}(Q, \widetilde{\text{SK}}):}$

Case 1: if $\exists(Q, \widetilde{\text{SK}}, K_0, K_1) \in T_{\text{EOT}_{\text{B}_2}}$,
return K_0, K_1;
Case 2: if $\exists(\text{SK}, 0, \widetilde{\text{PK}}, K_0) \in T_{\text{EOT}_{\text{A}_2}}$ and $(K_1, m_1, C_1) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$ and $\exists(Q, m_0, m_1, \widetilde{\text{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\text{PK}}, e, C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\text{SK}})$,
return K_0, K_1;
Case 3: if $\exists(\text{SK}, 1, \widetilde{\text{PK}}, K_1) \in T_{\text{EOT}_{\text{A}_2}}$ and $(K_0, m_0, C_0) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$ and $\exists(Q, m_0, m_1, \widetilde{\text{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\text{PK}}, e, C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where

$w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$,

return K_0, K_1;

Case 4: if $\exists(K_0, m_0, C_0) \wedge (K_1, m_1, C_1) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$ and $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widehat{\mathrm{PK}}, e, C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$ where $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$,

return K_0, K_1;

Case 5: otherwise, randomly sample K_0, K_1, $T_{\mathrm{EOT}_{\mathrm{B}_2}} = T_{\mathrm{EOT}_{\mathrm{B}_2}} \cup (Q, \widetilde{\mathrm{SK}}, K_0, K_1)$;

return K_0, K_1;

<u>$\mathcal{S}^{P_0}(\widehat{\mathrm{PK}}, e, C_0, C_1)$:</u>

Case 1: if $\exists(\widehat{\mathrm{PK}}, e, C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,

return w;

Case 2: if $\exists(K_0, m_0, C_0) \wedge (K_1, m_1, C_1) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$ and $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, \widetilde{\mathrm{PK}}) \in T_{\mathrm{EOT}_{\mathrm{B}_1}}$ and $\exists(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}}) \in T_{\mathcal{E}_1} \cup T_{\mathcal{E}_1^{-1}}$,

query $\mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$, obtain w;

return w;

Case 3: otherwise, randomly sample w, $T_{P_0} = T_{P_0} \cup (\widehat{\mathrm{PK}}, e, C_0, C_1, w)$;

return w.

<u>$\mathcal{S}^{P_0^{-1}}(w)$:</u>

Case 1: if $\exists(\widehat{\mathrm{PK}}, e, C_0, C_1, w) \in T_{P_0} \cup T_{P_0^{-1}}$,

return $(\widehat{\mathrm{PK}}, e, C_0, C_1)$;

Case 2: if $\exists(K_0, m_0, C_0) \wedge (K_1, m_1, C_1) \in T_{\mathcal{E}_2} \cup T_{\mathcal{E}_2^{-1}}$ and $\exists(Q, m_0, m_1, \widetilde{\mathrm{SK}}, e) \in T_{H_0}$ and $\exists(\widetilde{\mathrm{SK}}, \widetilde{\mathrm{PK}}) \in T_{\mathrm{EOT}_{\mathrm{B}_1}}$ and $\exists(e, \widetilde{\mathrm{PK}}, \widehat{\mathrm{PK}}) \in T_{\mathcal{E}_1} \cup T_{\mathcal{E}_1^{-1}}$, such that $w = \mathsf{I.OT}_2(Q, m_0, m_1, \widetilde{\mathrm{SK}})$;

return $(\widehat{\mathrm{PK}}, e, C_0, C_1)$;

Case 3: otherwise, randomly sample $(\widehat{\mathrm{PK}}, e, C_0, C_1)$, $T_{P_0^{-1}} = T_{P_0^{-1}} \cup (\widehat{\mathrm{PK}}, e, C_0, C_1, w)$;

return $(\widehat{\mathrm{PK}}, e, C_0, C_1)$.

The detailed proof process is very similar to that of Theorem 3.