Indifferentiability for Public Key Cryptosystems

Mark Zhandry[1] and Cong Zhang[2]

[1] Princeton University & NTT Research, mzhandry@princeton.edu
[2] Rutgers University, cz200@cs.rutgers.edu

Abstract. We initiate the study of indifferentiability for public key encryption and other public key primitives. Our main results are definitions and constructions of public key cryptosystems that are indifferentiable from ideal cryptosystems, in the random oracle model. Cryptosystems include:

- Public key encryption;
- Digital signatures;
- Non-interactive key agreement.

Our schemes are based on standard public key assumptions. By being indifferentiable from an ideal object, our schemes automatically satisfy a wide range of security properties, including any property representable as a single-stage game, and can be composed to operate in higher-level protocols.

Keywords. Indifferentiability, Composition, Public key encryption, Random oracle model, Ideal cipher model.

1 Introduction

When designing a cryptographic system, it is difficult to predict how it will be used in practice and what security properties will be required of it. For example, if the larger system produces certain error messages, this can lead to chosen ciphertext attacks [6]. Perhaps a message is encrypted using random coins which themselves are derived from the message, as is used for de-duplication [30]. Maybe the secret key itself will be encrypted by the system, as is sometimes used in disk encryption. Or perhaps there was bad randomness generation on the hardware device, leading to secret keys or encryption randomness that is low-entropy or correlated across many instances.

Cryptographers have devised different security models to capture each of the scenarios above and more, each requiring different constructions to satisfy. However, seldom are these different security models considered in tandem, meaning that each application scenario may require a different scheme. Even worse, there are many potential security models that have yet to be considered; after all, it is difficult to predict the various applications devised by software developers that may deviate from the existing provably secure uses.

With the above in mind, our goal is to develop a *single* construction for a given cryptographic concept that simultaneously captures any reasonable security property and can be composed to work in any reasonable larger protocol. As such, only a single instance of the scheme needs to be developed and then deployed in a variety of use cases, even those that have not been discovered yet.

Ideal Hash Functions: The Random Oracle Model. Our inspiration will be the random oracle model (ROM) [4], a common heuristic used in cryptography. Here, a hash function is assumed to be so well designed that the only reasonable attacks simply evaluate the hash function as a black box and gain nothing by trying to exploit the particular design. To capture this, the hash function is modeled as a truly random function, accessible by making queries to the function.

A random oracle truly is the "ideal" hash function: it is trivially one-way and collision resistant, the standard security notions for hash functions. But it is also much stronger: it is correlation intractable [9], a good extractor even for computational sources, and much more. When used in a larger system, random oracles can yield provably secure schemes even when standard security properties for hash functions are insufficient. In fact, the most efficient schemes in practice are often only known to be secure using random oracles. As such, the ROM is ubiquitous in cryptography.

Other idealized models have been studied before. Examples include the the ideal cipher model [27], the generic group model [28], and more recently ideal *symmetric* key encryption [2]. However, no prior work considers idealized models for public key cryptosystems.

Ideal Public Key Cryptosystems. In this work, we define and construct the first ideal *public key* cryptosystems such as public key encryption and digital signatures. By being ideal, our schemes will immediately satisfy a wide class of security properties, including most studied in the literature. Our schemes will be proven to be ideal in the random oracle model using Maurer's indifferentiability framework [22], under general computational assumptions. We also show that certain classic relations among cryptographic objects also hold in the ideal setting, while discussing cases where such relations fail.

Our goal comes with interesting challenges: on one hand, public key schemes tend to require number-theoretic structure in order to attain the necessary functionality. On the other hand, ideal schemes by definition have essentially no structure. Therefore, our results require novel techniques, including bringing indifferentiability into the public key setting.

1.1 What Is An Ideal Public Key Scheme?

Now, we turn to our results. Our first result is to define, precisely, what an "ideal" public key cryptosystem is. For simplicity, in the following discussion, we will consider the case of two-party non-interactive key exchange (NIKE). Such a scheme consists of two algorithms. KEYGEN is a key generation algorithm run by each of two users. We will adopt the convention that the input to KEYGEN is the user's secret key SK, and the output is the corresponding public key PK. The two users then exchange their public keys. They then run $\mathsf{SHAREDKEY}$ to extract a common shared key. $\mathsf{SHAREDKEY}$ will take as input the public key for one user and the secret key for the other, and output a shared key K. The correctness requirement is that both users arrive at the same key:

$$\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2) = \mathsf{SHAREDKEY}(\mathsf{PK}_2, \mathsf{SK}_1)$$

whenever $\mathsf{PK}_1 = \mathsf{KEYGEN}(\mathsf{SK}_1)$ and $\mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2)$.

Any NIKE scheme will have the syntax above and the same correctness requirement. On the other hand, any given NIKE scheme may have additional structural properties that make it insecure in certain settings. For example, if multiple shared keys are generated using the same public key PK for a given user, the resulting shared keys may be correlated in some algebraic way. In order to be secure in the widest variety of settings, an ideal NIKE scheme should therefore not have any such additional structure over the minimum needed to ensure correctness.

In the case of existing idealized models, the idealization is simply a uniformly random choice of procedures subject to the mandatory correctness requirements. For example, a hash function has no correctness requirement except for determinism; as such its idealization is a random oracle. Likewise, a block cipher must be a (keyed) permutation, and the decryption functionality must be its inverse. As such, the ideal cipher is a random keyed permutation and its inverse.

Therefore, the natural way to model an ideal NIKE scheme is to have all algorithms be random functions. Of course, the correctness requirement means that there will be correlations between the algorithms. We take an ideal NIKE scheme to be two oracles KEYGEN, $\mathsf{SHAREDKEY}$ such that:

- $\mathsf{KEYGEN}(\mathsf{SK})$ is a random injection;
- $\mathsf{SHAREDKEY}(\mathsf{PK}, \mathsf{SK})$ is a random function, except that $\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2) = \mathsf{SHAREDKEY}(\mathsf{PK}_2, \mathsf{SK}_1)$ whenever $\mathsf{PK}_1 = \mathsf{KEYGEN}(\mathsf{SK}_1)$ and $\mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2)$ [3].

We emphasize that all functions are public and visible to the attacker and the formal definition for ideal NIKE is given in section 3.1.

1.2 Indifferentiability

Of course, just like a random oracle/generic group/ideal cipher, ideal NIKE cannot exist in the real world. This then begs the question: how do we design and rigorously argue that a NIKE scheme is so well designed that it can be treated as an ideal NIKE scheme in applications?

Barbosa and Farshim [2] offer one possible answer. They build a symmetric key encryption scheme from a hash function. Then, they show, roughly, that if the hash function is ideal (that is, a random oracle), then so is their encryption scheme. Our goal in this work will be to do the same for public key schemes: to build an ideal NIKE scheme assuming that a hash function H is a random oracle.

As in [2], formal justification of ideal security requires some care. Suppose we have a construction of an ideal NIKE scheme $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ in the random oracle model, meaning each of the algorithms makes queries to a random function H. In the case where H is *hidden* to the adversary, such a construction is

[3] By the injectivity of KEYGEN, this is still a well-defined function.

almost trivial, and would essentially reduce to building symmetric key encryption from a PRF. However, Maurer [22] observed that this is *not* enough, since H is a *public* function and the adversary can query H as well.

Clearly, any construction $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ will now be distinguishable from the idealized algorithms, since the adversary can evaluate the algorithms for himself by making queries to H, and checking if the results are consistent with the oracles provided. Instead, what is needed is Maurer's notion of *indifferentiability*, which says that when $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ are ideal, it is possible to *simulate* H by a simulator S which can make queries to $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$. In the real world, $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ are constructed from H per the specification. In the ideal world, $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ are the idealized objects, and H is simulated by making queries to the ideal objects. Indifferentiability requires that the two worlds are indistinguishable to an adversary that gets access to all the oracles.

Maurer shows that indifferentiability has many desirable properties: it composes well and will be *as good as* the ideal object in many settings (see Section 1.3 below for some limitations).

Therefore, our goal will be to build NIKE which is indifferentiable from ideal NIKE in the random oracle model. As indifferentiability has mostly been used in the symmetric key setting, this will require new techniques to bring indifferentiability into the public key world. Indeed, most works on indifferentiability build ideal objects with minimal correctness requirements: none in the case of random oracles, and bijectivity/injectivity in the case of ideal ciphers/symmetric key encryption. The case of public key cryptosystems requires significantly more structure for correctness. In fact, we face an immediate theoretical barrier: Impagliazzo and Rudich[19] demonstrate that a random oracle is incapable of constructing something as structured as public key encryption, even ignoring the strong indifferentiability requirement.

Instead, we will obtain our needed structure using public key tools. However, public key tools come with *too much* structure: every term has an algebraic meaning which is not present in the idealized setting. Therefore, our goal will actually be to employ a novel combination of public key techniques *together with* random oracles in order to eliminate this extra structure. The result will be an indifferentiable NIKE scheme.

1.3 Discussion

Limitations. Before giving our constructions in detail, we briefly discuss limitations. Most importantly, idealized cryptosystems do not exist in the real world. Even more, Canetti, Goldreich, and Halevi [9] demonstrate that no concrete instantiation in the standard model is "as good as" an ideal object. Therefore, idealizations of cryptographic primitives are only heuristic evidence for security.

Nevertheless, the counter-examples are usually somewhat contrived, and do not apply in typical real-world settings. Indeed, in the case of hash functions, significant resources have been invested in analyzing their security, and the best

attacks typically treat the hash function as a random oracle[4]. As such, the random oracle appears to be a reasonable approximation to the real world in most settings. By building schemes from such strong hash functions and proving security using indifferentiability, such schemes are essentially "as good as" the ideal schemes, assuming the underlying hash function is ideal.

In fact, the random oracle model is widely used for the construction of new cryptosystems, as it allows security to be justified where no obvious concrete security property for hash functions would suffice. In these cases, the system is typically proven to satisfy the single security property considered. In our case, we are able to rely on the same heuristic treatment of hash functions, and attain ideal security.

Now, Ristenpart, Shacham, and Shrimpton [24] demonstrate the limitations of the indifferentiability framework. In particular, they show that indifferentiability is *insufficient* for proving security properties defined by *multi-stage* games. While this potentially precludes certain applications, indifferentiability is still sufficient to prove many security properties such as CCA-security, key-dependent-message and circular security in restricted settings (see [2] for discussion), bounded leakage resilience, and more. We also note that if all but one of the stages are independent of the ideal primitives, then indifferentiability is sufficient. This captures, for example, the usual modeling of deterministic public key encryption in the random oracle model [3]. Even more, in the case where multiple stages depend on the ideal primitives, Mittelbach [23] shows that indifferentiability is sufficient in some settings.

We leave as an interesting direction for future work building ideal public key schemes that can be proven secure in stronger models of indifferentiability such as reset indifferentiability [24] or context-restricted indifferentiability [21].

Relationship to Universal Composability. There are some parallels between our idealized cryptosystems and *universal composability* (UC) [8]. Both seek to define an "ideal" object for a given cryptographic concept. Both consider composition, using one ideal functionality to build another. Both use a simulation to define security and composition.

However, the two notions are also fundamentally different. In UC, an ideal functionality is specified by considering how a trusted third party would solve a given cryptographic task. Then the actual protocol "emulates" this ideal through an interactive protocol. For example, the cryptographic task solved by encryption is simply private communication, and as such the ideal functionality is more or less message passing.

In contrast, in our setting, rather than consider the abstract task such as message passing, we consider the cryptographic abstraction of solving such a task concretely. So we consider, for example, public key encryption rather than the task of private communication. The ideal public key encryption scheme is then a random choice of functions, subject to the constraints imposed by the

[4] Of course, there are several exceptions, for instance, the length-extension attacks against various Merkle-Damgard-based hash functions, such as SHA-2 or MD5.

correctness requirements. The main benefit of our approach is that we naturally handle a much wider set of use cases such as key-dependent message security and weak/non-existent randomness; there is no natural way to model these use cases in the UC framework.

1.4 Constructing Ideal NIKE

We now turn to our constructions. Our goal will be to combine a *standard model* NIKE $(\mathsf{keygen}, \mathsf{sharedkey})$ — one with concrete mild security properties that are easy to instantiate — with random oracles to obtain an ideal model NIKE $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$.

Making KEYGEN *indifferentiable.* First, we will focus just on KEYGEN, which on its own must be indifferentiable from a random injection. Of course, we could just set KEYGEN to be a random oracle[5], but we want to somehow incorporate keygen so that it can provide the structure needed when we turn to construct SHAREDKEY. Nevertheless, a random oracle (or some other idealized object) is needed somewhere in the construction. As a first attempt, we could consider defining $\mathsf{KEYGEN}(\mathsf{SK}) = H(\mathsf{keygen}(\mathsf{SK}))$, hashing the output of keygen to eliminate any structure on the public keys. This, unfortunately, does not work. For example, keygen may not be collision resistant, and any collision for keygen will therefore give a collision for KEYGEN. The resulting KEYGEN would then clearly be distinguishable from a random function without even making queries to H.

Attack 1. Even if we assume keygen was collision resistant, the scheme would still not be indifferentiable. Indeed, the attacker can query $\mathsf{KEYGEN}(\mathsf{SK})$, evaluate $\mathsf{pk} = \mathsf{keygen}(\mathsf{SK})$ for itself, and then query H on pk. The simulator now has to simulate H, and for indifferentiability to hold it must know how to set $H(\mathsf{pk}) = \mathsf{KEYGEN}(\mathsf{SK})$. However, the simulator only gets to see pk and somehow must query KEYGEN on SK. Extracting SK from pk involves breaking the original NIKE, which is presumably intractable.

A different approach would be to define $\mathsf{KEYGEN}(\mathsf{SK}) = \mathsf{keygen}(H(\mathsf{SK}))$. The problem here is that keygen may output very structured public keys, which are clearly distinguishable from random. One possibility is to assume keygen has pseudorandom public keys; that is, that keygen applied to uniformly random coins gives a pseudorandom output.

Attack 2. However, we still have a problem. Indeed, suppose the adversary queries $\mathsf{KEYGEN}(\mathsf{SK})$, which in the ideal world will give a random string. Then the adversary queries $H(\mathsf{SK})$. In the ideal world, the simulator must set $H(\mathsf{SK}) = r$ such that $\mathsf{keygen}(r) = \mathsf{KEYGEN}(\mathsf{SK})$. This may be flat out impossible (in the case where the range of keygen is sparse), and at a minimum requires inverting keygen, again breaking the security of the NIKE scheme.

A third approach which does work is to combine the two: $\mathsf{KEYGEN}(\mathsf{SK}) = H_1(\mathsf{keygen}(H_0(\mathsf{SK})))$. Now both H_0, H_1 are random oracles that are simulated by

[5] By having the random oracle be sufficiently expanding, it will be an injection with high probability.

the simulator. This actually gives indifferentiability: when the adversary queries $H_0(\mathsf{SK})$, the simulator will program $H_0(\mathsf{SK}) = r$ for a randomly chosen r. Then it will program $H_1(\mathsf{keygen}(r)) = \mathsf{KEYGEN}(\mathsf{SK})$ by querying KEYGEN. The only way a problem can arise is if the input $\mathsf{keygen}(r)$ was already programmed in H_1. All that we need to exclude such a possibility is that keygen is well-spread: that the distribution of outputs given a uniformly random input has high min-entropy. This follows easily from the security of the NIKE protocol.

The takeaway from the above discussion is that inputs and outputs for a standard-model scheme must be processed by idealized objects; this is the only way that the simulator can obtain enough information to be indifferentiable.

Making $\mathsf{SHAREDKEY}$ *indifferentiable.* Next, we move to define $\mathsf{SHAREDKEY}$ in a way to make the joint oracles $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ indifferentiable from an ideal NIKE protocol.

Unfortunately, we immediately run into problems. We somehow need to design the shared-key algorithm $\mathsf{SHAREDKEY}$ to take as input one public key $\mathsf{PK}_1 = H_1(\mathsf{keygen}(H_0(\mathsf{SK}_1)))$, as well as another secret key SK_2. It will output a shared key K. Importantly, we need to maintain the correctness requirement that $\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2) = \mathsf{SHAREDKEY}(\mathsf{PK}_2, \mathsf{SK}_1)$ whenever $\mathsf{PK}_1 = \mathsf{KEYGEN}(\mathsf{SK}_1)$ and $\mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2)$.

Guided by Impagliazzo and Rudich's [19] barrier, we cannot rely on the functionalities of the random oracles H_0, H_1 for this. Instead, we must use the functionality provided by $(\mathsf{keygen}, \mathsf{sharedkey})$. However, $\mathsf{sharedkey}$ expects output from keygen, and this value has been completely scrambled by the hash function H_1, which is un-invertible. Therefore, $\mathsf{SHAREDKEY}$ has no way to apply $\mathsf{sharedkey}$ in a meaningful way.

So we need some way to preserve the structure of the output keygen while still allowing for an indifferentiability proof. But at the same time, we cannot just expose the output of keygen in the clear, as explained above.

Our solution is to replace H_1 with a random *permutation* P such that *both* P and P^{-1} are publicly accessible (we discuss instantiating the random permutation below). We then have that

$$\mathsf{KEYGEN}(\mathsf{SK}) = P^{-1}(\mathsf{keygen}(H_0(\mathsf{SK})))$$

Then we can define $\mathsf{SHAREDKEY}(\mathsf{PK}, \mathsf{SK}) = \mathsf{sharedkey}(P(\mathsf{PK}), H_0(\mathsf{SK}))$. Note that, defining $\mathsf{SHAREDKEY}$ in this way achieves the desired correctness guarantee, which follows simply from the correctness of $(\mathsf{keygen}, \mathsf{sharedkey})$.

Attack 3. However, by allowing the permutation P to be invertible, we have invalidated our indifferentiability proof above for KEYGEN. Suppose for example that keygen's outputs are easily distinguishable from random. Then an attacker can compute $\mathsf{PK} = \mathsf{KEYGEN}(\mathsf{SK})$, and then query P on either PK or a random string r. In the case of a random string, $P(r)$ will itself be essentially a random string. On the other hand, $P(\mathsf{PK})$ will be an output of keygen, and hence distinguishable from random. The problem is that the simulator defining P^{-1} only gets to see r and has no way to know whether r came from KEYGEN or was

just a random string. Therefore, the attacker can fool the simulator, leading to a distinguishing attack.

To avoid this problem, we will assume the standard-model NIKE protocol has *pseudorandom public keys*. In this case, the simulator will *always* respond to a P using a fresh random output of keygen. In the case where the query to P was on a random r, the result will look random to the adversary. On the other hand, if the query was on a $\mathsf{PK} = \mathsf{KEYGEN}(\mathsf{SK})$, the simulator is ready to program any subsequent $H_0(\mathsf{SK})$ query to satisfy $P(\mathsf{PK}) = \mathsf{keygen}(H_0(\mathsf{SK}))$.

Attack 4. Many more problems still arise, similar to the problems above faced when trying to define $\mathsf{KEYGEN}(\mathsf{SK}) = \mathsf{keygen}(H(\mathsf{SK}))$. Namely, the adversary could first call the query $k = \mathsf{SHAREDKEY}(\mathsf{PK}', \mathsf{SK})$, which in the ideal world will give a random string. Then the adversary makes queries to P, H_0 and computes $\mathsf{sharedkey}(P(\mathsf{PK}'), H_0(\mathsf{SK}))$ for itself. In the ideal world, the simulator must set $P(\mathsf{PK}') = r$ and $H_0(\mathsf{SK}) = s$ such that $\mathsf{sharedkey}(r, s) = k$. But this involves inverting $\mathsf{sharedkey}$ on k, which may be computationally infeasible. Worse yet, the adversary could do this for $\mathsf{PK}'_1, \ldots, \mathsf{PK}'_\ell$ and $\mathsf{SK}_1, \ldots, \mathsf{SK}_\ell$, obtaining ℓ^2 different random and independent $k_{i,j}$ values from $\mathsf{SHAREDKEY}$ by considering all possible $\mathsf{PK}'_i, \mathsf{SK}_j$ pairs. The simulator then needs to somehow find $r_1, \ldots, r_\ell, s_1, \ldots, s_\ell$ such that $\mathsf{sharedkey}(r_i, s_j) = k_{i,j}$, where $k_{i,j}$ are each random independent strings. This is clearly impossible for large enough ℓ, since it would allow for compressing an $O(\ell^2)$-bit random string into $O(\ell)$ bits.

Our solution is to apply one more hash function, this time to the output of $\mathsf{sharedkey}$: $\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1(\mathsf{sharedkey}(P(\mathsf{PK}_1), H_0(\mathsf{SK}_2)))$. Now, all we need is that $\mathsf{sharedkey}(r_i, s_j)$ are all distinct for different i, j pairs, which follows with high probability from the security of the NIKE scheme. Then we can simply program $H_1(\mathsf{sharedkey}(r_i, s_j)) = k_{i,j}$.

Attack 5. This construction unfortunately is still insecure: the adversary first samples sk_1 and SK_2, then it queries $\mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2), \mathsf{pk}_2 = P(\mathsf{PK}_2)$, then calculates $k = \mathsf{sharedkey}(\mathsf{pk}_2, \mathsf{sk}_1)$ and queries $H_1(k)$. After that, the adversary calculates $\mathsf{pk}_1 = \mathsf{keygen}(\mathsf{sk}_1)$, and queries $\mathsf{PK}_1 = P^{-1}(\mathsf{pk}_1)$. Next it calls $k' = \mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2)$ and finally tests $k' \stackrel{?}{=} H_1(k)$. In the real world, the test always passes, while to achieve indifferentiability, the simulator has to output a proper $H_1(k)$. Unfortunately, until the query $H_1(k)$, simulator knows nothing of $(\mathsf{sk}_1, \mathsf{SK}_2)$ (it only has $\mathsf{KEYGEN}(\mathsf{SK}_2)$), which means that it cannot program $H_1(k)$ to be k'. Therefore test fails with overwhelming probability in the ideal world. To get prevent this attack, we present our final construction:

$$\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1(\{\mathsf{PK}_1, \mathsf{PK}_2\}, \mathsf{sharedkey}(P(\mathsf{PK}_1), H_0(\mathsf{SK}_2)))$$

where $\mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2) = P^{-1}(\mathsf{keygen}(H_0(\mathsf{SK}_2)))$[6]. How does this help? We note that, in the final construction, by including $\mathsf{PK}_1, \mathsf{PK}_2$ in the H_1 queries, we force the adversary to query $\mathsf{PK}_1 = P^{-1}(\mathsf{pk}_1)$ before the H_1 query. This allows the simulator to program P^{-1} in a way that allows it to correctly answer the

[6] Here, $\{\mathsf{PK}_1, \mathsf{PK}_2\}$ means the un-ordered set containing PK_1 and PK_2, so that $\{\mathsf{PK}_1, \mathsf{PK}_2\} = \{\mathsf{PK}_2, \mathsf{PK}_1\}$.

later H_1 query. In particular, it samples SK_1 itself and responds to the P^{-1} query with $\mathsf{PK}_1 = \mathsf{KEYGEN}(\mathsf{SK}_1)$. Afterward, when then adversary makes the query $H_1(\{\mathsf{PK}_1, \mathsf{PK}_2\}, k)$, the simulator will respond with $\mathsf{SHAREDKEY}(\mathsf{PK}_2, \mathsf{SK}_1)$, which is always identical to $\mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2)$.

Attack 6. Even with our final construction, we must be careful. Suppose that it was possible for the adversary to choose a public key pk^* such that it can guess the value of $\mathsf{sharedkey}(\mathsf{pk}^*, \mathsf{sk})$ for a random (hidden) sk, then there is *still* an attack. Namely, the adversary queries $\mathsf{PK}^* = P^{-1}(\mathsf{pk}^*)$ and $\mathsf{PK} = \mathsf{KEYGEN}(\mathsf{SK})$ for a random SK. It then guesses the value t^* of $\mathsf{sharedkey}(\mathsf{PK}^*, H_0(\mathsf{SK}))$, without ever actually querying H_0. Finally, it queries $k = H_1(\{\mathsf{PK}^*, \mathsf{PK}\}, t^*)$, and checks that the result is equal to $\mathsf{SHAREDKEY}(\mathsf{PK}^*, \mathsf{SK})$. In the real world, this check will pass as long as the guess t^* is correct. On the other hand, in the ideal world, the check can only pass with negligible probability, since the simulator has no way of knowing SK, and hence has no way of programming H_1 to output $\mathsf{SHAREDKEY}(\mathsf{PK}^*, \mathsf{SK})$. Our solution is to add another security requirement for the NIKE scheme, which we call *entropic shared keys*, insisting that for any pk^* of the adversary's choosing, the adversary *cannot* guess $\mathsf{sharedkey}(\mathsf{pk}^*, \mathsf{sk})$ except for negligible probability.

Attack 7. One last attack strategy: the attacker could first query $\mathsf{PK}_1 = \mathsf{KEYGEN}(\mathsf{SK}_1), \mathsf{PK}_2 = \mathsf{KEYGEN}(\mathsf{SK}_2), k = \mathsf{SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2)$. Then, it could query $r_1 = P(\mathsf{PK}_1), r_2 = P(\mathsf{PK}_2)$. Finally, it could treat r_1, r_2 as the messages in the standard-model NIKE protocol, and guess the shared key t for the protocol. Then it could query H_1 on t (and $\{\mathsf{PK}_1, \mathsf{PK}_2\}$). In the real world, the result $H_1(\{\mathsf{PK}_1, \mathsf{PK}_2\}, t)$ would be equal to k, so the simulator in the ideal world needs to be able to set $H_1(\{\mathsf{PK}_1, \mathsf{PK}_2\}, t) = k$. At this point, the simulator has $\mathsf{PK}_1, \mathsf{PK}_2, t$. But the simulator has no knowledge of SK_1 or SK_2. Therefore it has no way of guessing the correct input to $\mathsf{SHAREDKEY}$ to obtain k, as doing so requires recovering either SK_1 or SK_2 by inverting the random injection KEYGEN. Moreover, the adversary has access to both honest interfaces and adversarial interfaces, and it can implement those interfaces into two oracles: $\mathcal{O}_1(\cdot, \cdot)$ and $\mathcal{O}_2(\cdot, \cdot)$, such that $\mathcal{O}_i$ takes $(\mathsf{pk}, \mathsf{shk})$ as inputs and outputs "1", iff shk is a valid shared key of r_i and pk (pk here is allowed to be a malicious public key). Concretely, take $\mathcal{O}_1$ for instance, the adversary can implement the oracle as follows: let $\mathsf{PK}^* = P^{-1}(\mathsf{pk})$, $\bar{k} = H_1(\mathsf{PK}^*, \mathsf{PK}_1, \mathsf{shk})$ and $k^* = \mathsf{SHAREDKEY}(\mathsf{PK}^*, \mathsf{SK}_1)$, and we define $\mathcal{O}_1(\mathsf{pk}, \mathsf{shk}) = 1$ if $\bar{k} = k^*$[7]. Note that, if our NIKE scheme is only against passive attack, such as the scheme in [20], then the active attack [18], which only uses the oracles described above, can efficiently decode the secret key of r_1 and r_2, and break indifferentiability. Thus, to prevent such kind of attacks, we need a higher level security notion for our standard model NIKE, which we call *semi-active unpredictable shared keys*. In this notion, we define that, even having access to $\mathcal{O}_1$ and $\mathcal{O}_2$, guessing the shared key t from the user's messages r_1, r_2 is implausible except for a negligible probability.

[7] As long as the range of H_1 is sufficiently large, the oracle implemented only has negligible false-positive.

While we have protected against certain natural attacks, we need to argue indifferentiability against all possible attacks. To do so we use a careful simulation strategy for H_0, H_1, P, P^{-1}, and prove indifferentiability through a careful sequence of hybrids. In essence, each step in the hybrid argument corresponds roughly to one of the attack strategies discussed above, and our proof shows that these attacks do not work, demonstrating the indistinguishability of the hybrids.

Constructing P, P^{-1}. Our random permutation P, P^{-1} can easily be instantiated using the ideal cipher model in the setting where the key space contains only a single element. We note that indifferentiable ideal ciphers can be constructed from random oracles [12].

Therefore, all we need for our construction is three random oracles. Multiple random oracles can easily be built from a single random oracle by prefixing the oracle index to the input. Finally, an indifferentiable random oracle of any domain/range can be constructed from a fixed-size random oracle [10].

The role of P in the proof. It is natural to wonder what role P plays in the actual security of our scheme. After all, since P^{-1} is publicly invertible using P, the adversary can easily undo the application of P^{-1} to the output of KEYGEN. So it may seem that P is a superfluous artifact of the proof.

There are multiple ways to address this question. One answer is that without P, there would be no way to have a computationally efficient simulator as discussed above. One could consider an inefficient simulator, but this would correspond to a weaker notion of indifferentiability. This notion of indifferentiability would be useless when composing with protocols that have computational rather than statistical security. What's more, we would actually be unable to prove even this weaker form. Indeed, our proof crucially relies on the computational security of the standard-model NIKE protocol. Since the inefficient simulator would essentially have to break the security of the NIKE protocol, it would be impossible to carry out the proof.

A higher-level answer is that by including P — which is under full control of the simulator — the simulator gets to learn extra information about what values the adversary is interested in. In particular, in order to relate the ideal oracles to the standard-model scheme, the adversary must always send a query to the simulator. This extra information provided by making such queries is exactly what the simulator needs for the proof to go through. This is a common phenomenon in random-oracle proofs, where hashing sometimes has no obvious role except to provide a reduction/simulator with necessary information.

Yet another answer is that, if P is omitted, the scheme is actually insecure in some settings. For example, an ideal NIKE satisfies the property that an adversary, given Alice's secret key and *half* of Bob's public key, cannot compute the shared key between Alice and Bob. Now, consider the case where the standard model NIKE does *not* satisfy this requirement. Then if we do not include P, our construction does not satisfy the requirement either. Instead, by including P, an adversary who gets half of Bob's ideal public key cannot invert the permutation

to recover *any* information about the corresponding standard-model public key. It then follows that the adversary cannot guess the shared key.

1.5 Extending to Other Idealized Cryptosystems

We now turn our attention to extending the above results to other cryptosystems. First, we use our ideal NIKE scheme to construct ideal public key encryption (PKE). Note that ideal public key encryption is in particular CCA secure, whereas the standard way to turn a NIKE scheme into a PKE scheme is never CCA secure. In order to make the scheme CCA secure, a natural starting point is the Fujisaki-Okamoto (FO) transform [17]. While this transformation applied to our ideal NIKE certainly achieves CCA security, it unfortunately is not indifferentiable when applied to our NIKE. The reasons are several-fold, and should come as no surprise given that FO was never designed to achieve indifferentiability.

For starters, recall from our NIKE discussion that all inputs and outputs of the algorithms need to be passed through ideal objects under the simulator's control. In the FO transformation, this is not the case. Another reason why FO does not give indifferentiability is that the FO transform allows for encryption randomness to be recovered during decryption; in fact, this is a crucial feature of the CCA security proof. On the other hand, such encryption schemes cannot be ideal, since ideal encryption schemes guarantee that the encryption randomness is hidden even to the decrypter[8].

To overcome these issues, we first show a careful transformation from our ideal NIKE into ideal *deterministic* public key encryption (DPKE). By focusing first on DPKE, we side-step the randomness issue. Our transformation is inspired by the FO transform, but in order to ensure that all inputs/outputs are passed through oracles under the simulator's control, we employ our random permutation trick again.

Finally, we turn to convert ideal DPKE into standard PKE. The usual conversion (simply including the encryption randomness as part of the message) does not suffice, again because the usual conversion allows the decrypter to recover the encryption randomness. We instead essentially hide the randomness by hashing with a random oracle. This, however, requires care in order to enable a complete indifferentiability proof.

Ideal Signatures. Finally, we investigate constructing ideal signatures. While in the standard model signatures can in principle be built from one-way functions and therefore random oracles, we observe that the situation for ideal signatures is much more challenging. For example, an ideal signature scheme will be *unique*, meaning for any message/public key, only a single signature will verify. On the other hand, constructing unique signatures even under standard security notions

[8] One can define a different idealization of PKE where the ideal decryption functionality *does* output the encryption randomness. However, this stronger functionality corresponds to weaker security guarantees

is difficult, and the only known constructions require strong number-theoretic tools such as bilinear maps.

We instead assume a building block as a standard-model signature scheme with unique signatures, as well as some other mild security properties which can be easily instantiated using bilinear maps. We show that such a scheme can, in fact, be turned into ideal signatures using similar ideas to the above.

1.6 Instantiations

Our NIKE schemes require a standard-model NIKE. Unfortunately, we cannot use a truly arbitrary standard model NIKE, as in addition to the semi-active unpredictable shared keys, we also need pseudorandom public keys and entropic shared keys. As such, we need to make sure such a scheme can be instantiated. Our other results similarly require standard-model schemes where various outputs of the schemes are pseudorandom bit strings.

We note that the entropic shared key requirement is satisfied by all constructions we are aware of and the semi-active unpredictable shared keys can be done under doubly-strong CDH assumption or bilinear maps [16]. On the other hand, the requirement of pseudorandom public keys is slightly non-trivial. Many number-theoretic constructions have public keys that are elements in $\mathbb{Z}_q^k$ for some modulus k; even if the public keys are pseudorandom in these sets, there may be no way to represent a random element of $\mathbb{Z}_q^k$ as a random bit string (which we need in order to apply the ideal permutation P), since $q^k = |\mathbb{Z}_q^k|$ may be far from a power of 2.

However, it will usually be easy to map such public keys to random strings in $\{0,1\}^n$ for some integer n. For example, in the case $k = 1$, suppose we are given a (pseudo)random element $\mathsf{pk} \in \mathbb{Z}_q$. Let n be some integer such that $n \geq \lambda + \log_2 q$ for a security parameter λ. Let $t = \lfloor 2^n/q \rfloor$ be the largest integer such that $tq \leq 2^n$. Then we can extend pk into a random element pk' in $\mathbb{Z}_{tq}$ by setting $\mathsf{pk}' = \mathsf{pk} + aq$, where a is a random integer in $\mathbb{Z}_t$. Finally, we note that a random integer in $\mathbb{Z}_{tq}$ is distributed exponentially close (in λ) to a random integer in $\mathbb{Z}_{2^n}$.

We can similarly handle the case $k > 1$ by bijecting public keys into $\mathbb{Z}_{q^k}$ in the standard way. Such conversions can be applied to Diffie-Hellman key agreement. The result that we attain our NIKE results under doubly-strong-CDH, whereas our signature scheme requires CDH in bilinear map groups.

We note that one of the cases we do not know how to handle are schemes based on factoring or RSA, as the public key would be an RSA composite $p \times q$ for random large primes p and q; this is clearly distinguishable from a random string using primality testing. On the other hand, we know of no way to bijectively map such numbers into random bit strings, without factoring them (and hence breaking the scheme). In fact, doing so without factoring would seem to allow for *obliviously* sampling large RSA composites by choosing a random string, and then applying the inverse map. Oblivious sampling of RSA composites is a major open question.

2 Background

NOTATION. Throughout this paper, $\lambda \in \mathbb{N}$ denotes the security parameter. We let $\mathbb{N}$ be the set of non-negative integers, including zero and $\{0,1\}^*$ denote the set of all finite-length bit strings, including the empty string ϵ ($\{0,1\}^0 = \epsilon$). For two bit strings, X and Y, $X||Y$ denotes string concatenation and (X,Y) denotes a uniquely decodable encoding of X and Y. The length of a string X is denoted by $|X|$.

For a finite set $\mathcal{S}$, we denote $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For a probabilistic algorithm A, we denote $y \leftarrow A(x;R)$ the process of running A on inputs x and randomness R, and assigning y the result. We let $\mathcal{R}_A$ denote the randomness space of A; we require $\mathcal{R}_A$ to be the form $\mathcal{R}_A = \{0,1\}^r$. We write $y \leftarrow A(x)$ for $y \leftarrow A(x,R)$ with uniformly chosen $R \in \mathcal{R}_A$, and we write $y_1, \ldots, y_m \leftarrow A(x)$ for $y_1 \leftarrow A(x), \ldots, y_m \leftarrow A(x)$ with fresh randomness in each execution. If A's running time is polynomial in λ, then A is called probabilistic polynomial-time (PPT). We say a function $\mu(n)$ is negligible if $\mu \in o(n^{-\omega(1)})$, and is non-negligible otherwise. We let $\mathsf{negl}(n)$ denote an arbitrary negligible function. If we say some $p(n)$ is poly, we mean that there is some polynomial q such that for all sufficiently large n, $p(n) \leq q(n)$. We say a function $\rho(n)$ is noticeable if the inverse $1/\rho(n)$ is poly. We use boldface to denote vector, i.e. $\boldsymbol{m}$; we denote $\boldsymbol{m}_i$ as the i-th component of $\boldsymbol{m}$ and $|\boldsymbol{m}|$ as the length of $\boldsymbol{m}$. Due to space limit, we will give the definition of games and ideal objects(such as random oracle model, ideal cipher model) in Appendix A.

2.1 Public Key Primitives

In this part, we recall the definitions of the public key primitives that we consider in our work.

NON-INTERACTIVE KEY EXCHANGE (NIKE) [13]. NIKE is a cryptographic primitive which enables two parties, who know the public keys of each other, to agree on a symmetric shared key without requiring any interaction. It consists of two algorithms: $\mathsf{NIKE.keygen}$ and $\mathsf{NIKE.sharedkey}$ together with a shared key space $\mathcal{SHK}$.

- $\mathsf{NIKE.keygen}$: Given input a secret key sk, the algorithm outputs a public key pk;
- $\mathsf{NIKE.sharedkey}$ Given inputs a public key $\mathsf{pk_1}$ and a secret key $\mathsf{sk_2}$, the algorithm outputs a shared key $\mathsf{shk} \in \mathcal{SHK}$.

For correctness, we require that, for any two key pairs $(\mathsf{pk_1}, \mathsf{sk_1}), (\mathsf{pk_2}, \mathsf{sk_2})$, the system satisfies:

$$\mathsf{NIKE.sharedkey}(\mathsf{pk_1}, \mathsf{sk_2}) = \mathsf{NIKE.sharedkey}(\mathsf{pk_2}, \mathsf{sk_1}).$$

PUBLIC KEY ENCRYPTION (PKE) [13]. A public-key encryption scheme consists of three algorithms: $\mathsf{PKE.keygen}, \mathsf{PKE.enc}, \mathsf{PKE.dec}$ together with a message space $\mathcal{M}$. Formally,

- $\mathsf{PKE.keygen}$ Given input a secret key sk, the algorithm outputs a public key pk;
- $\mathsf{PKE.enc}$ Given inputs a public key pk and $m \in \mathcal{M}$, the algorithm outputs a ciphertext $\mathsf{c} = \mathsf{PKE.enc}(\mathsf{pk}, \mathsf{m})$;
- $\mathsf{PKE.dec}$ Given inputs a secret key sk and a ciphertext c, the algorithm outputs either a plaintext m or $\perp$.

For correctness, we require that, for any key pair $(\mathsf{pk}, \mathsf{sk})(\mathsf{pk} = \mathsf{PKE.keygen}(\mathsf{sk}))$ and $\mathsf{m} \in \mathcal{M}$, the scheme satisfies:

$$\mathsf{PKE.dec}(\mathsf{sk}, \mathsf{PKE.enc}(\mathsf{pk}, \mathsf{m})) = \mathsf{m}.$$

Digital Signature [25]. A digital signature scheme consists of three algorithms: $\mathsf{Sig.keygen}, \mathsf{Sig.sign}, \mathsf{Sig.ver}$ along with a message space $\mathcal{M}$. Formally,

- $\mathsf{Sig.keygen}$ Given input a sign key sk, the algorithm outputs a verification key vk;
- $\mathsf{Sig.sign}$ Given inputs a sign key sk and a message $\mathsf{m} \in \mathcal{M}$, the algorithm outputs a signature $\sigma = \mathsf{Sig.sign}(\mathsf{sk}, \mathsf{m})$;
- $\mathsf{Sig.ver}$ Given inputs a signature σ, a message and a verification key vk, outputs either 1 or 0.

For correctness, we require that, for any key pair $(\mathsf{sk}, \mathsf{vk})$ $(\mathsf{vk} = \mathsf{Sig.keygen}(\mathsf{sk}))$ and $\mathsf{m} \leftarrow \mathcal{M}$, the signature scheme satisfies:

$$\mathsf{Sig.ver}(\mathsf{Sig.sign}(\mathsf{sk}, \mathsf{m}), \mathsf{m}, \mathsf{vk}) = 1$$

2.2 Indifferentiability

In [22], Maurer, Renner and Holenstein(MRH) propose the indifferentiability framework, which formalizes a set of necessary and sufficient conditions for one system to securely be replaced with another one in a wide class of environments. This framework has been used to prove the structual soundness of a number of cryptographic primitives, which includes hash functions [10, 14], blockciphers [1, 12, 15], domain extenders [11] and authenticated encryption with associated data [2]. In the following, we first recall the definition of indifferentiability.

A random system $\Sigma := (\Sigma.\mathrm{hon}, \Sigma.\mathrm{adv})$ is accessible via two interfaces $\Sigma.\mathrm{hon}$ and $\Sigma.\mathrm{adv}$, where $\Sigma.\mathrm{hon}$ provides a honest interface through which the system can be accessed by all parties and $\Sigma.\mathrm{adv}$ models the adversarial access to the inner working part of Σ. Typically, a system implements either some ideal objects $\mathcal{F}$, or a construction $C^{\mathcal{F}'}$, which applies some underlying ideal objects $\mathcal{F}'$.

Definition 1. (**Indifferentiability** [22].) *Let Σ_1 and Σ_2 be two systems 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*

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

where games $\mathrm{Real}_{\Sigma_1,\mathcal{D}}$ *and* $\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}$ *are defined in Figure 1. 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 weakly indifferentiable from* Σ_2*, if for any probabilistic polynomial time differentiator* $\mathcal{D}$*, there exists an efficient simulator* $\mathcal{S}_{\mathcal{D}}$ *such that the advantage above is negligible.*

$\underline{\mathrm{Real}_{\Sigma_1,\mathcal{D}}:}$	$\underline{\textsc{Honest}(X)}$	$\underline{\mathrm{Ideal}_{\Sigma_2,\mathcal{D}}:}$	$\underline{\textsc{Proc.Const}(X)}$
$b \leftarrow \mathcal{D}^{\textsc{Honest},\textsc{Adv}}$,	Return $\Sigma_1.\mathrm{hon}(X)$.	$b \leftarrow \mathcal{D}^{\textsc{Honest},\textsc{Adv}}$,	Return $\Sigma_2.\mathrm{hon}(X)$.
Return b.	$\underline{\textsc{Adv}(X)}$	Return b.	$\underline{\textsc{Proc.Prim}(X)}$
	Return $\Sigma_1.\mathrm{adv}(X)$.		Return $\mathcal{S}^{\Sigma_2.\mathrm{adv}(\cdot)}(X)$.

Fig. 1: Indifferentiability of Σ_1 and Σ_2, where $\mathcal{S}$ is the simulator and $\mathcal{D}$ is the adversary.

Next, we recall composition theorem for indifferentiability. In [22], MRH give out the composition theorem for indifferentiability, and then Ristenpart, Shacham and Shrimpton(RSS) [24] propose a game-based version for the theorem.

Theorem 2. (**Indifferentiability Composition** [24].) *Let* $\Sigma_1 := (\mathrm{C}^{\mathcal{F}_1}, \mathcal{F}_1)$ *be a system that is indifferentiable from* $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ *along with simulator* $\mathcal{S}$*. Let* $\mathcal{G}$ *be a single-stage game. Then for any adversary* $\mathcal{A}$*, there exists an adversary* $\mathcal{B}$ *and a differentiator* $\mathcal{D}$ *such that*

$$\Pr[\mathcal{G}^{\mathrm{C}^{\mathcal{F}_1},\mathcal{A}^{\mathcal{F}_1})}] \leq \Pr[\mathcal{G}^{\mathcal{F}_2,\mathcal{B}^{\mathcal{F}_2})}] + \mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}.$$

However, RSS [24] prove that the composition theorem above does not extend to multi-stage games as the simulator has to keep the local state for consistency. While, Barbosa and Farshim(BF) [2] observe that if allowing some relaxations on the games, we could rewrite some multi-stage games as equivalent to single-stage games. Essentially, for an n-adversary game $\mathcal{G}_n^{C^{\mathcal{F}},\mathcal{A}_1,\ldots,\mathcal{A}_n}$, if only one adversary(say $\mathcal{A}_1$) can call the ideal objects $\mathcal{F}$ directly and the rest can only call $C^{\mathcal{F}}$, then $\mathcal{G}_n$ can be rewritten as a single-stage game, because the game $\mathcal{G}_n$ itself, of course, has access to $C^{\mathcal{F}}$. Then in [2], BF formalize this observation in the following theorem.

Theorem 3. (**Multi-stage Game Composition** [2].) *Let* $\Sigma_1 := (\mathrm{C}^{\mathcal{F}_1}, \mathcal{F}_1)$ *be a system that is indifferentiable from* $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ *along with simulator* $\mathcal{S}$*. Let* $\mathcal{G}$ *be an* n*-adversary game and* $\mathcal{A} := (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ *be a* n*-tuple of adversaries where* $\mathcal{A}_1$ *can access* $\mathcal{F}_1$ *but* $\mathcal{A}_i$ *(*$i > 1$*) can only access* $\mathrm{C}^{\mathcal{F}_1}$*. Then there is an* n*-adversary* $\mathcal{B}$ *and a differentiator* $\mathcal{D}$ *such that*

$$\Pr[\mathcal{G}^{\mathrm{C}^{\mathcal{F}_1},\mathcal{A}_1^{\mathcal{F}_1},\mathcal{A}_2^{\mathrm{C}^{\mathcal{F}_1}},\ldots,\mathcal{A}_n^{\mathrm{C}^{\mathcal{F}_1}}}] = \Pr[\mathcal{G}^{\mathcal{F}_2,\mathcal{B}_1^{\mathcal{F}_2},\ldots,\mathcal{B}_n^{\mathcal{F}_2}}] + \mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}.$$

Remark. Barbosa and Farshim [2] give a strong motivation for the relaxation imposed on the class of games above. To our best of knowledge, the related-key

attack (key-dependent message attack) game is not known to be equivalent to any single-stage game. As a result, it would be insufficient to prove a system is related-key attack secure as follows: 1) there is another system, say Σ_2, such that Σ_1 is indifferentiable from Σ_2; 2) Σ_2 is related-key attack secure. However, if allowing the relaxation, the proof follows trivially, hence from a practical point of view(by adding this specific relaxation on games), composition extends well beyond 1-adversary games.

3 Indifferentiable NIKE

In this section, we propose the notion of "ideal NIKE" and then build an indifferentiable non-interactive key exchange scheme from simpler ideal primitives and a standard-model NIKE scheme.

3.1 What is Ideal NIKE?

In this part we give the rigorous description of ideal NIKE, formally:

Definition 4. (**Ideal NIKE.**) *Let $\mathcal{X}, \mathcal{Y}, \mathcal{W}$ be three sets such that $|\mathcal{X}| \geq 2^{\omega(\log \lambda)}$, $|\mathcal{Y}| \geq 2^{\omega(\log \lambda)}, |\mathcal{W}| \geq 2^{\omega(\log \lambda)}$, $|\mathcal{X}| \leq |\mathcal{Y}|$ and $|\mathcal{X}| \times |\mathcal{Y}| \leq |\mathcal{W}|$. We denote $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$ as the set of all injections that map $\mathcal{X}$ to $\mathcal{Y}$ and $\mathcal{G}[\mathcal{X} \times \mathcal{Y} \to \mathcal{W}]$ as the set of the functions that map $\mathcal{X} \times \mathcal{Y}$ to $\mathcal{W}$. We define $\mathcal{T}$ as the set of all function pairs (F, G) such that: 1)$F \in \mathcal{F}, G \in \mathcal{G}$; 2) $\forall x, y \in \mathcal{X}, G(x, F(y)) = G(y, F(x))$; 3) $G(x_1, y_1) = G(x_2, y_2) \Rightarrow (x_1 = x_2 \wedge y_1 = y_2) \vee (y_1 = F(x_1) \wedge y_2 = F(x_2))$.*

We say that a NIKE scheme $\Pi_{\mathsf{NIKE}} = (\mathsf{NIKE.KEYGGEN}, \mathsf{NIKE.SHAREDKEY})$, associated with secret key space $\mathcal{X}$, public key space $\mathcal{Y}$ and shared key space $\mathcal{Z}$, is an ideal NIKE if $(\mathsf{NIKE.KEYGEN}, \mathsf{NIKE.SHAREDKEY})$ is sampled from $\mathcal{T}$ uniformly.

It's trivial to note that, due to an information-theoretic argument, an ideal NIKE achieves related-key attack security, leakage-resiliency and so forth. Next, we show how to construct an indifferentiable NIKE scheme from simpler primitives.

3.2 Construction

In this section, we build an indifferentiable NIKE scheme from simpler ideal primitives (namely random oracles and ideal ciphers) along with a standard-model (that is, *non-ideal*) NIKE scheme.

Building Blocks. Our scheme consists of several building blocks:

- A standard-model NIKE scheme $\Pi_{\mathsf{SM-NIKE}} = (\mathsf{keygen}, \mathsf{sharedkey})$ with secret key space $\mathcal{X}$, public key space $\mathcal{Y}$, and shared key space $\mathcal{Z}$;
- $H_0 := \{0,1\}^* \to \mathcal{X}$ is a random oracle whose co-domain matches the secret key space of Π.

- $H_1 := \{0,1\}^* \to \mathcal{W}$ is a random oracle, where $|\mathcal{X}| \times |\mathcal{Y}| \leq |\mathcal{W}|$;
- $P := \mathcal{Y} \to \mathcal{Y}$ is a random permutation on the public key space of Π, and P^{-1} is P's inverse.

Note that, the random permutations typically operate on bit strings, which means $\mathcal{Y} = \{0,1\}^n$ for some natural number $n \geq \omega(\log \lambda)$. Moreover, the shared key space in the standard model NIKE $\mathcal{Z}$ and in our construction $\mathcal{W}$ might be not equivalent, because it's unnecessarily correct that $|\mathcal{X}| \times |\mathcal{Y}| \leq |\mathcal{Z}|$. And if not, then setting $\mathcal{W} = \mathcal{Z}$ would give a differentiator directly, by just checking whether $|\mathcal{X}| \times |\mathcal{Y}| \leq |\mathcal{W}|$.

Construction. Now we are ready to build an indifferentiable NIKE scheme, denoted as $\Pi_{\mathsf{NIKE}} = (\mathsf{NIKE.KEYGEN}, \mathsf{NIKE.SHAREDKEY})$, from the building blocks above. Formally,

- $\mathsf{NIKE.KEYGEN}(\mathsf{SK})$: Given input SK, the algorithm runs $\mathsf{keygen}(H_0(\mathsf{SK}))$, and outputs the public key $\mathsf{PK} = P^{-1}(\mathsf{keygen}(H_0(\mathsf{SK})))$;
- $\mathsf{NIKE.SHAREDKEY}(\mathsf{PK}_1, \mathsf{SK}_2)$: Given inputs $(\mathsf{PK}_1, \mathsf{SK}_2)$, the algorithm computes $\mathsf{PK}_2 = \mathsf{NIKE.KEYGEN}(\mathsf{SK}_2)$ and $\mathsf{sharedkey}(P(\mathsf{PK}_1), H_0(\mathsf{SK}_2))$. If $\mathsf{PK}_1 \leq \mathsf{PK}_2$, then it outputs the shared key as

$$\mathsf{SHK} = H_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{sharedkey}(P(\mathsf{PK}_1), H_0(\mathsf{SK}_2))),$$

  else, it outputs

$$\mathsf{SHK} = H_1(\mathsf{PK}_2, \mathsf{PK}_1, \mathsf{sharedkey}(P(\mathsf{PK}_1), H_0(\mathsf{SK}_2))).$$

Correctness of the scheme easily follows, and what's more interesting is its indifferentiability. Next, we prove our scheme is indifferentiable from an ideal NIKE. Before that, we first specify the security properties of the standard-model NIKE.

Property 1. Semi-active unpredictable shared key. We say the shared key, for a NIKE scheme, is semi-active unpredictable, if for any PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}_{\mathcal{A}} := \Pr[\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(\mathsf{pk}_1, \mathsf{pk}_2) = \mathsf{sharedkey}(\mathsf{pk}_1, \mathsf{sk}_2)] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{pk}_i = \mathsf{keygen}(\mathsf{sk}_i), \mathsf{sk}_i \leftarrow \mathcal{X}$ and $\mathcal{O}_i$ is a predicate oracle such that takes $(\mathsf{pk}, \mathsf{shk})$ as input and outputs a bit(the public key pk here might be malicious). Concretely, the oracle $\mathcal{O}_i$ outputs "1" iff $\mathsf{shk} = \mathsf{sharedkey}(\mathsf{sk_i}, \mathsf{pk})$. This is the standard security game for NIKE schemes against active adversary, except that we relax the notion on two pieces: 1) we only require unpredictability of the shared key, rather than indistinguishability from random; 2) the oracles takes both public key pk and shared key shk as input and tell whether shk is a valid shared key, rather than taking the public key pk, and outputting the corresponding shared key shk.

Remark. Note that, unpredictable shared key only against passive attack might be insufficient for indifferentiability. Take the scheme in [20] for instance, its

shared key is, of course, unpredictable against passive adversary. However, the active attack in [18] can decode the secret keys (sk_1 and sk_2) efficiently, by only accessing to the oracles above, which breaks the indifferentiability immediately.

The next two properties are mild additional security properties that are not usually required for NIKE schemes, but are achieved by most natural schemes. We require these properties in order to prove the ideal security of our construction.

Property 2. ENTROPIC SHARED KEYS. We say the shared key, for a NIKE scheme is entropic, if for any PPT adversary $\mathcal{A}$, the advantage that $\mathcal{A}$ wins the following game is negiligble:

$\mathsf{sk} \xleftarrow{\$} \mathcal{X}, (\mathsf{pk}^*, \mathsf{shk}^*) \leftarrow \mathcal{A};$
Return $1((\mathsf{shk}^* = \mathsf{sharedkey}(\mathsf{pk}^*, \mathsf{sk}))$.

Fig. 2: Entropic Shared Keys.

Note that the entropic shared keys property tells us that if the adversary only knows one public key(even it's chosen by the adversary), it cannot predict the shared key if the other secret key is random and hidden. In other words, this property guarantees that there is no way to make the shared key have low min-entropy.

Property 3. PSEUDORANDOM PUBLIC KEYS. We say the public key, for a NIKE scheme, is pseudorandom, if for any PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}_{\mathcal{A}} := |\Pr[\mathcal{A}(\mathsf{keygen}(\mathsf{sk}))] - \Pr[\mathcal{A}(R)]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{sk} \leftarrow \mathcal{X}, R \leftarrow \mathcal{Y}$. We immediately observe that as $\mathcal{Y} = \{0,1\}^n$, our standard-model NIKE must have public keys that are pseudorandom bit strings.

Additional Property. Based on entropic shared keys and pseudorandom public keys, we actually have an additional security property, which we call *variant entropic shared keys* (this property is necessary for the proof below), specifically:

$$\Pr[\mathsf{shk}^* = \mathsf{sharedkey}(\mathsf{pk}, \mathsf{sk}^*) : (\mathsf{sk}^*, \mathsf{shk}^*) \leftarrow \mathcal{A}, \mathsf{pk} \xleftarrow{\$} \mathcal{Y}] \leq \mathsf{negl}(\lambda),$$

where now the adversary is asked to output a secret key sk^* and a shared key shk^* with a random and hidden public key pk. This property holds trivially based on Property 2 and 3. In fact, if there is an attacker breaks this variant entropic shared keys with non-negligible probability ρ, by outputting $(\mathsf{sk}^*, \mathsf{shk}^*)$, then we can break property 2, by just outputting $(\mathsf{keygen}(\mathsf{sk}^*), \mathsf{shk}^*)$, with probability $\rho - \epsilon$, where ϵ is the advantage of pseudorandom public keys. By definition, we have that $\Pr_{\mathsf{pk} \xleftarrow{\$} \mathcal{Y}}[\mathsf{shk}^* = \mathsf{sharedkey}(\mathsf{sk}^*, \mathsf{pk})] \geq \rho$, hence, due to pseudorandom public keys, it's apparent that

$$\Pr_{\mathsf{sk} \xleftarrow{\$} \mathcal{X}}[\mathsf{shk}^* = \mathsf{sharedkey}(\mathsf{sk}^*, \mathsf{keygen}(\mathsf{sk})) = \mathsf{sharedkey}(\mathsf{sk}, \mathsf{keygen}(\mathsf{sk}^*))] \geq \rho - \epsilon,$$

which breaks property 2 immediately. Below, for ease of exposition, we abuse this variant property also as *entropic shared keys*. And we say a NIKE scheme is Good if it satisfies the three properties above.

Now, we are ready to establish our theorem.

Theorem 5. *(***(Indifferentiable NIKE).***)* Π_{NIKE} *is indifferentiable from an ideal NIKE if* $\Pi_{\mathsf{SM-NIKE}}$ *is* Good.

Proof. According to the definition of indifferentiability, we immediately observe that the adversary has two honest interfaces ($\mathsf{NIKE.KEYGEN}$, $\mathsf{NIKE.SHAREDKEY}$) (below we will denote ($\mathsf{NKG}, \mathsf{NSK}$) for ease) and four adversarial interfaces (H_0, P, P^{-1}, H_1). Therefore, we need to build an efficient simulator $\mathcal{S}$ that can simulate the four adversarial interfaces H_0, P, P^{-1} and H_1 properly, which means, 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, there exists a system that responds to all of the queries(both honest and adversarial) in a slightly different way and then we build our simulator $\mathcal{S}$ as the system in the last game. Before the description of the games, we first specify some parameters and events:

- there are six types of query: $(\mathsf{SK}; H_0), (\mathsf{PK}; P), (\mathsf{pk}; P^{-1}), (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$, $(\mathsf{SK}; \mathsf{NKG}), (\mathsf{PK}, \mathsf{SK}; \mathsf{NSK})$ where $\mathsf{SK} \leftarrow \mathcal{X}, \mathsf{PK}, \mathsf{pk}, \mathsf{PK}_1, \mathsf{PK}_2 \leftarrow \mathcal{Y}, \mathsf{shk} \leftarrow \mathcal{Z}$;
- adversary makes at most q queries to the system, where $q = \mathsf{poly}(\lambda)$;
- the oracles used in the real world are $\tilde{H}_0, \tilde{P}, \tilde{P}^{-1}, \tilde{H}_1, \widetilde{\mathsf{NKG}}$ and $\widetilde{\mathsf{NSK}}$;
- in each game, the system's responses are denoted as $H_0^{\mathsf{r}}, P^{\mathsf{r}}, P^{-1\mathsf{r}}, H_1^{\mathsf{r}}, \mathsf{NKG}^{\mathsf{r}}$ and $\mathsf{NSK}^{\mathsf{r}}$, for instance, we denote $H_0^{\mathsf{r}}(\mathsf{SK})$ as the system's response when adversary makes a query $Q = (\mathsf{SK}; H_0)$;
- the advantage of *unpredictable shared key* is bounded by ϵ_1;
- the advantage of *entropic shared keys* is bounded by ϵ_2;
- the advantage of *pseudorandom public keys* is bounded by ϵ_3.

Note that in the normal lazy sampling of random oracles or permutations, each output will be chosen essentially at random. However, a simulator for indifferentiability will need to occasionally sample in such a way as to be consistent with the ideal NIKE. Next we define two events, named "Strong Consistency Check"(SCC) and "Weak Consistency Check"(WCC). These checks look for the cases where the adversary may already have the necessary information to predict the query response without making the query. Intuitively, if the checks pass($\mathsf{SCC} = 1$), it means the adversary is unable to make such a prediction, and the simulator is essentially free to answer randomly. On the other hand, if the checks fail($\mathsf{SCC} = 0$), the simulator must answer carefully to be consistent with the adversary's prediction.

Strong Consistency Check. Let $Q_1, \ldots, Q_q$ be the sequence of the queries, we say the check for the k-th query passes ($\mathsf{SCC}_k = 1$) if any one of the following cases is satisfied:

- Case 1. The k-th query is a P query, say $(\mathsf{PK}^*; P)$ and in the previous $k-1$ queries, there is no query with form of $(\mathsf{SK}^*; H_0), (\mathsf{SK}^*; \mathsf{NKG})$ or $(\mathsf{PK}, \mathsf{SK}^*; \mathsf{NSK})$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$.
  Note that if there had been a previous query on such an SK^*, then it would be possible for the adversary to predict $P(\mathsf{PK}^*) = P(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*))$ without making a P query at all by just evaluating $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*))$.
- Case 2. The k-th query is a P^{-1} query, say $(\mathsf{pk}^*; P^{-1})$ and in the previous $k-1$ queries, there exists no query with form of $(\mathsf{SK}^*; H_0), (\mathsf{SK}^*; \mathsf{NKG})$ or $(\mathsf{PK}, \mathsf{SK}^*; \mathsf{NSK})$ such that $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*)) = \mathsf{pk}^*$.
  Note that if there had been a previous query on such an SK^*, then the adversary can predict $P^{-1}(\mathsf{pk}^*) = P^{-1}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*)))$ by just querying $(\mathsf{SK}^*; \mathsf{NKG})$.
- Case 3. The k-th query is a H_1 query, say $(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}; H_1)$ which satifies $\mathsf{PK}_1^* > \mathsf{PK}_2^*$ or in the previous $k-1$ queries, then is no query with form of $(\mathsf{SK}^*; H_0), (\mathsf{SK}^*; \mathsf{NKG})$ or $(\mathsf{PK}, \mathsf{SK}^*; \mathsf{NSK})$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) \in \{\mathsf{PK}_1^*, \mathsf{PK}_2^*\}$.
  Note that if there had been a previous query on such an SK^* (say $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}_1^*$) and $\mathsf{PK}_1^* \leq \mathsf{PK}_2^*$, then the adversary can predicate $H_1(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk})$ by just querying $(\mathsf{PK}_2^*, \mathsf{SK}^*; \mathsf{NSK})$[9].

We note, in the ideal world, the simulator is unable to tell if SCC happens since doing so requires knowing the adversary's queries to NKG and NSK. Instead, the simulator will be able to carry out a weak consistency check, WCC:

Weak Consistency Check. Let $Q_1, \ldots, Q_q$ be the sequence of the queries, we say event WCC occurs for the k-th query ($\mathsf{WCC}_k = 1$) if one of the following cases satisfies:

- Case 1. The k-th query is a P query, say $(\mathsf{PK}^*; P)$ and in the previous $k-1$ queries, there is no query with form of $(\mathsf{SK}^*; H_0)$ such that $\mathsf{NKG}^{r}(\mathsf{SK}^*) = \mathsf{PK}^*$.
- Case 2. The k-th query is a P^{-1} query, say $(\mathsf{pk}^*; P^{-1})$ and in the previous $k-1$ queries, there exists no query with form of $(\mathsf{SK}^*; H_0)$ such that $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*)) = \mathsf{pk}^*$.
- Case 3. The k-th query is a H_1 query, say $(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}; H_1)$ which satifies $\mathsf{PK}_1^* > \mathsf{PK}_2^*$ or in the previous $k-1$ queries, then is no query with form of $(\mathsf{SK}^*; H_0)$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) \in \{\mathsf{PK}_1^*, \mathsf{PK}_2^*\}$.

Note that in the weak consistency check, we only keep track of the H_0 queries, rather than the NKG or NSK queries, so it will not catch all the bad cases. But we will demonstrate that the adversary is unable to generate such bad cases except for negligible probability.

Now we are ready to describe the games. After each game, we give the intuition for why that game is indistinguishable from the previous game. The full proof of indistinguishability between the hybrids will be given in Appendix E.

[9] This, of course, only works under the condition that shk is a valid shared key.

Game 0. This game is identical to the real game except that the system maintains four tables, referring to H_0-table, P-table, P^{-1}-table and H_1-table. Specifically, the system responds to the queries the same as in the real world, for instance, $H_0^{\mathsf{r}}(\mathsf{SK}) = \tilde{H}_0(\mathsf{SK}), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{NKG}}(\mathsf{SK})$ and so forth. For the tables, the system maintains them as follows:

- H_0-table: Initially empty, consists of tuples with form of $(\mathsf{SK}, \mathsf{sk}, \mathsf{pk}, \mathsf{PK})$. Once the adversary makes a H_0 query, say $(\mathsf{SK}^*; H_0)$, which does not exist in H_0-table (no tuple that the first element of it is SK^*), inserts $(\mathsf{SK}^*, \tilde{H}_0(\mathsf{SK}^*), \mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)), \widetilde{\mathsf{NKG}}(\mathsf{SK}^*))$ into H_0-table.
- P-table: Initially empty, consists of tuples with form of $(*, *, \mathsf{pk}, \mathsf{PK})$. Once the adversary makes a P query, say $(\mathsf{PK}^*; P)$, which does not exist in P-table, it inserts $(*, *, \tilde{P}(\mathsf{PK}^*), \mathsf{PK}^*)$ into P-table.
- P^{-1}-table: Initially empty, consists of tuples with form of $(*, *, \mathsf{pk}, \mathsf{PK})$. Once the adversary makes a P^{-1} query, say $(\mathsf{pk}^*; P^{-1})$, which does not exist in P^{-1}-table, it inserts $(*, *, \mathsf{pk}^*, \tilde{P}^{-1}(\mathsf{pk}^*))$ into P^{-1}-table.
- H_1-table: Initially empty, consists of tuples with form of $(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}, \mathsf{SHK})$. Once the adversary makes a H_1 query, say $(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}^*; H_1)$, which does not exist in H_1-table, the system inserts $(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}^*, \tilde{H}_1(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}^*))$ into H_1-table.

Note that at this point all the queries are responded by the real oracles and these tables are just keeping track of information related to adversary's queries(to the adversarial interfaces) and completely hidden to the adversary. Hence the view in real game is identical to the one in Game 0. Next, we illustrate an alternative way to responds to part of the queries, by using these tables and accessing to the honest interfaces.

For ease of exposition, we here define a relation between the query Q and the table Tab. Specifically, if Q is a H_0 query ($Q = (\mathsf{SK}; H_0)$), we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ in Tab such that $T_1 = \mathsf{SK}$. Analogously, if Q is a P query ($Q = (\mathsf{PK}; P)$), we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ in Tab such that $T_4 = \mathsf{PK}$; if Q is a P^{-1} query ($Q = (\mathsf{pk}; P^{-1})$), we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ in Tab such that $T_3 = \mathsf{pk}$; if Q is a H_1 query ($Q = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$), we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ in Tab such that $T_1 = \mathsf{PK}_1, T_2 = \mathsf{PK}_2$ and $T_3 = \mathsf{shk}$.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries. Specifically,

H_0-QUERY. Suppose $Q_k = (\mathsf{SK}; H_0)$ ($k \in [1, q]$), then the system responds as follows:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{SK}$, then the system responds with T_2;
- Case 2. If $Q_k \notin H_0$, the system makes a query $(\mathsf{SK}; \mathsf{NKG})$ and searches every tuple $T = (T_1, T_2, T_3, T_4) \in P$ such that $T_4 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ and $T_2 \neq *$. If such a tuple is found, then the system responds to Q_k with T_2 and inserts $(\mathsf{SK}, T_2, T_3, T_4)$ into H_0-table.

– Case 3. Otherwise, the system responds with $\tilde{H}_0(\mathsf{SK})$ and inserts $(\mathsf{SK}, \tilde{H}_0(\mathsf{SK}), \mathsf{keygen}(\tilde{H}_0(\mathsf{SK})), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0-table.

P-QUERY. Suppose $Q_k = (\mathsf{PK}; P)$, then the system responds:

– Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P \cup P^{-1}$ such that $T_4 = \mathsf{PK}$, then the system responds with T_3;
– Case 2. Otherwise, responds with $\tilde{P}(\mathsf{PK})$, and inserts $(*, *, \tilde{P}(\mathsf{PK}), \mathsf{PK})$ into both P and P^{-1} table.

P^{-1}-QUERY. Suppose $Q_k = (\mathsf{pk}; P^{-1})$, then the system responds:

– Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P \cup P^{-1}$ such that $T_3 = \mathsf{pk}$, then the system responds with T_4;
– Case 2. Otherwise, responds with $\tilde{P}^{-1}(\mathsf{pk})$, and inserts $(*, *, \mathsf{pk}, \tilde{P}^{-1}(\mathsf{pk}))$ into both P and P^{-1} table.

H_1-QUERY. Suppose $Q_k = (\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk}; H_1)$, then the system responds:

– Case 1: If $Q_k \in H_1$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ such that $T_1 = \mathsf{PK_1}, T_2 = \mathsf{PK_2}$ and $T_3 = \mathsf{shk}$, then the system responds with T_4.
  Note that for the cases below where $Q_k \notin H_1$, after we compute the response SHK, we will always add the tuple $(\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk}, \mathsf{SHK})$ to H_1-table, so that on future identical queries, Q_k will be in H_1.
– Case 2: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 1$, then the system would first test the validity of shk using P and P^{-1} table. Formally,
  1. If $\mathsf{PK_1} > \mathsf{PK_2}$, then the system responds with $\tilde{H}_1(\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk})$;
  2. If $\mathsf{PK_1} \leq \mathsf{PK_2}$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P^{-1}$ such that $T_4 = \mathsf{PK_1}$ and $T_1 \neq *$, then the system searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4) \in P \cup P^{-1}$ which satisfies $T'_4 = \mathsf{PK_2}$ and $T'_2 \neq *$. If such a tuple T' is found, then test if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the system makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with $\tilde{H}_1(\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk})$;
  3. If $\mathsf{PK_1} \leq \mathsf{PK_2}$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P^{-1}$ such that $T_4 = \mathsf{PK_2}$ and $T_1 \neq *$, then the system searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4) \in P \cup P^{-1}$ which satisfies $T'_4 = \mathsf{PK_1}$ and $T'_2 \neq *$. If such a tuple T' is found, then test if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the system makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with $\tilde{H}_1(\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk})$;
  4. Otherwise, responds with $\tilde{H}_1(\mathsf{PK_1}, \mathsf{PK_2}, \mathsf{shk})$.
– Case 3: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 0$, which means, there exists a query $Q_i = (\mathsf{SK}; H_0)(i < k)$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) \in \{\mathsf{PK_1}, \mathsf{PK_2}\}$, then the system would first test the validity of z using H_0-table, P-table and P^{-1}-table. Formally,

1. There exist two queries $Q_i = ((\mathsf{SK}_1; H_0))$ and $Q_j = (\mathsf{SK}_2; H_0)(i, j < k)$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_\ell) = \mathsf{PK}_\ell(\ell = 1, 2)$, then the system test $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(\mathsf{keygen}(H_0(\mathsf{SK}_1)), H_0(\mathsf{SK}_2))$(only using the tuples in H_0-table). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$, otherwise responds with $\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk})$;
2. There exists a query $Q_i = ((\mathsf{SK}_1; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}_1) = \mathsf{PK}_1$ but no such a query for PK_2, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_2$. If such a query is found, then the system tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}_1))(H_0(\mathsf{SK}_1)$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$, otherwise responds with $\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk})$;
3. There exists a query $Q_i = ((\mathsf{SK}_2; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}_2) = \mathsf{PK}_2$ but no such a query for PK_1, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_1$. If such a query is found, then the system tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}_2))(H_0(\mathsf{SK}_2)$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_1, \mathsf{SK}_2; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2)$, otherwise responds with $\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk})$;
4. Otherwise, responds with $\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk})$.

Note that, in Game 1 the system keeps a longer table, and for part of the queries, the system responds to them in an alternative way, which is *only* using the tables and accessing to the honest interfaces. Moreover, in Game 1, the tuples stored in the tables correspond to the response of queries that are answered by the real oracles, for instance, $\forall(\mathsf{SK}_1, \mathsf{PK}_2)$, we have $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1) = \widetilde{\mathsf{NKG}}(\mathsf{SK}_1)$ and $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1) = \widetilde{\mathsf{NSK}}(\mathsf{PK}_2, \mathsf{SK}_1)$. Hence, in either game, the response of any query is identical, which means that the view in Game 1 is identical to the one in Game 0. However, the system 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.

Remark: A careful reader would have noticed that several cases, for instance Case 2 in H_0-query or Case 2.2 in H_1-query, do not exist in Game 1, because for any tuple $T \in P \cup P^{-1}$, we have $T_1 = T_2 = *$. While, in the following hybrid games, the system would modify the way of maintaining the tables, and those values would not be $*$ anymore.

Game 2. This game is identical to Game 1, except for responding to P queries. Suppose $Q_k = (\mathsf{PK}; P)$, then the system responds:

- Case 1: If $Q_k \in H_0 \cup P \cup P^{-1}$, same as in Game 1.
- Case 2: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then the system responds with $y \leftarrow \mathcal{Y}$ and inserts $(*, *, y, \mathsf{PK})$ into P-table.

– Case 3: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$, then responds with $\tilde{P}(\mathsf{PK})$, and inserts $(*, *, \tilde{P}(\mathsf{PK}), \mathsf{PK})$ into both P and P^{-1} table.

The only difference between Game 1 and Game 2 occurs in the case 2, where $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$. In Game 1, the system responds with $\tilde{P}(\mathsf{PK})$ while in Game 2, we replace it with a random string. Due to definition, we note that strong consistency check determines whether or not the adversary has the ability to learn $\tilde{P}(\mathsf{PK})$ by making a query to NKG. If so, we must answer the P query using $\tilde{P}$; otherwise, we can answer it randomly without changing the view with high probability.

Game 3. This game is identical to Game 2, except modifying P-query once more. More concretely, in case 2, if $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then the system responds to Q_k with a random public key $\mathsf{pk}^* = \mathsf{keygen}(\mathsf{sk}^*)$ instead, where $\mathsf{sk}^* \xleftarrow{\$} \mathcal{X}$ and inserts $(*, *, \mathsf{keygen}(\mathsf{sk}^*), \mathsf{PK})$ into P-table.

Here, we just change the distribution of outputs in the case where we do not use $\tilde{P}$. Game 3 is indistinguishable from Game 2 by pseudorandom public keys.

Game 4. This game is identical to Game 3, except the way of filling up the P-table. Concretely, in case 2, the system inserts the tuple $(*, \mathsf{sk}^*, \mathsf{keygen}(\mathsf{sk}^*), \mathsf{PK})$ into the table.

Here, the only difference between Game 4 and Game 3 is that the system also records the random coins sk^* into the P-table(now the 2nd term of this tuple is not $*$ anymore). Since that sk^* is uniformly sampled by the system, it would not be used with high probability, which means the adversary's view would be close in both games.

Game 5. This game is identical to Game 4, except that to answer P queries. Assuming $Q_k = (\mathsf{PK}; P)$, then,

– Case 1. $Q_k \in H_0 \cup P \cup P^{-1}$, same as in Game 4.
– Case 2. If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then same as in Game 4.
– Case 3. If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$, then the system responds with a random public key $\mathsf{pk}^* = \mathsf{keygen}(\mathsf{sk}^*)$, where $\mathsf{sk}^* \xleftarrow{\$} \mathcal{X}$ and inserts $(*, \mathsf{sk}^*, \mathsf{keygen}(\mathsf{sk}^*), \mathsf{PK})$ into P-table.

The difference between Game 5 and Game 4 occurs in case 3, where the adversary makes a P query $Q_k = (\mathsf{PK}^*; P)$ such that $\mathsf{SCC}_k = 0$ and $Q_k \notin H_0 \cup P \cup P^{-1}$. This means that $\mathsf{PK}^* = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)))$ for some SK^*, and that the adversary previously queried $(\mathsf{SK}^*; \mathsf{NKG})$ or $(\mathsf{PK}, \mathsf{SK}^*; \mathsf{NSK})$, but not $(\mathsf{SK}^*; H_0)$.

Since $Q_k \notin H_0 \cup P \cup P^{-1}$, it's trivial that the adversary has never made a query on $(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)); P^{-1})$, as such a query would have resulted in a tuple which contains PK^* being added to P^{-1}-table. Therefore, $\tilde{H}_0(\mathsf{SK}^*)$ and thus $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ are independent of the adversary's view. However, in Game 4, the query would result in $(*, *, \tilde{P}(\mathsf{PK}), \mathsf{PK})$ being added to P-table, comparing to $(*, \mathsf{sk}^*, \mathsf{keygen}(\mathsf{sk}^*), \mathsf{PK}^*)$ in Game 5. Unlike the previous one, this time sk^* would be used, for instance, the adversary knows SK^* such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$,

and it would make a query $(\mathsf{SK}^*; H_0)$ after Q_k. While, by definition, the system in Game 5 implicitly sets $H_0^r(\mathsf{SK}^*) = \mathsf{sk}^*$, and sk^* is a fresh random value, hence the response of $(\mathsf{SK}^*; H_0)$ is well-distributed. Moreover, before Q_k, the adversary is independent of $\tilde{H}_0(\mathsf{SK}^*)$, which means the two games are indistinguishable.

Game 6. This game is identical to Game 5, except that to answer P^{-1} queries. Suppose $Q_k = (\mathsf{pk}; P^{-1})$, then the system responds:

- Case 1: If $Q_k \in H_0 \cup P \cup P^{-1}$, same as in Game 5;
- Case 2: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then the system samples $\mathsf{SK} \leftarrow \mathcal{X}$, makes a query $(\mathsf{SK}; \mathsf{NKG})$, responds to Q_k with $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ and inserts $(\mathsf{SK}, *, \mathsf{pk}, \mathsf{NKG}^r(\mathsf{SK}))$ into P^{-1}-table.
- Case 3. Otherwise, responds with $\tilde{P}^{-1}(\mathsf{pk})$.

The difference of Game 6 and Game 5 occurs in case 2, where the adversary makes a P^{-1} query $Q_k = (\mathsf{pk}^*; P^{-1})$ such that $\mathsf{SCC}_k = 1$ and $Q_k \notin H_0 \cup P \cup P^{-1}$. This means, there is no SK^* which appears in the previous queries, even in the NKG and NSK queries, such that $\mathsf{pk}^* = \mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$. Hence SK^* and thus $\widetilde{\mathsf{NKG}}(\mathsf{SK}^*)$ are independent of the adversary's view, which means we can replace $\widetilde{\mathsf{NKG}}(\mathsf{SK}^*)$ with $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ where $\mathsf{SK} \xleftarrow{\$} \mathcal{X}$ as long as $\mathsf{SK}, \tilde{H}_0(\mathsf{SK})$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}))$ do not appear in the adversary's queries(including both the previous queries and the following queries). In fact, SK is uniformly sampled by the system and independent of adversary's view, those values never appear except for negligible probability.

Game 7. This game is identical to Game 6, except that to answer P^{-1} queries. Suppose $Q_k = (\mathsf{pk}; P^{-1})$, then the system responds:

- Case 1: If $Q_k \in H_0 \cup P \cup P^{-1}$, same as in Game 6;
- Case 2: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, same as in Game 6.
- Case 3: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$, then the system samples $\mathsf{SK} \leftarrow \mathcal{X}$, makes a query $(\mathsf{SK}; \mathsf{NKG})$, responds to Q_k with $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ and inserts $(\mathsf{SK}, *, \mathsf{pk}, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into P^{-1}-table.

The difference between Game 7 and Game 6 occurs in case 3, where the adversary makes a P^{-1} query $Q_k = (\mathsf{pk}^*; P^{-1})$ such that $\mathsf{SCC}_k = 0$ and $Q_k \notin H_0 \cup P \cup P^{-1}$. This means that $\mathsf{pk}^* = \mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)) = \tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*))$ for some SK^*, and that the adversary previously queried $(\mathsf{SK}^*; \mathsf{NKG})$ or $(\mathsf{PK}, \mathsf{SK}^*; \mathsf{NSK})$, but not $(\mathsf{SK}^*; H_0)$.

Moreover, we note that in Game 6, the response of any P query$(Q \notin H_0)$ is responded with a random public key, rather than calling the real oracles, which means the query $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*); P)$ would leak nothing about $\tilde{H}_0(\mathsf{SK}^*)$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$. Hence, $\tilde{H}_0(\mathsf{SK}^*)$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ are hidden from the adversary's view. And if the adversary can make a query $(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)); P^{-1})$, then it means the adversary is able to predict the public key for an unknown secret key$(\tilde{H}_0(\mathsf{SK}^*))$. As our standard-model NIKE scheme has pseudorandom public keys, this is impossible except for negligible probability.

Game 8. This game is identical to Game 7, except that to answer H_0 queries. Suppose $Q_k = (\mathsf{SK}; H_0)$, then the system responds:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{SK}$, then the system responds with T_2(same as in Game 7);
- Case 2. If $Q_k \notin H_0$ but there is a tuple $T = (T_1, T_2, T_3, T_4)$ in P-table such that $T_4 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ and $T_2 \neq *$, then the system responds with T_2 and inserts $(\mathsf{SK}, T_2, T_3, T_4)$ into H_0-table(same as in Game 7);
- Case 3. Otherwise, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, makes a query $(\mathsf{SK}; \mathsf{NKG})$, responds with sk, and inserts $(\mathsf{SK}, \mathsf{sk}, \mathsf{keygen}(\mathsf{sk}), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0-table.

The difference between Game 8 and Game 7 occurs in case 3, where the adversary makes a H_0 query $Q_k = (\mathsf{SK}^*; H_0)$ such that $Q_k \notin H_0$ and $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*)$ never appears in P-table, hence it knows nothing of $\tilde{H}_0(\mathsf{SK}^*)$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$, although the adversary might know $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*)$. Thus, from the adversary's view, $\tilde{H}_0(\mathsf{SK}^*)$ is uniformly distributed in $\mathcal{X}$, and it is equivalent to randomly sample $\mathsf{sk}^* \leftarrow \mathcal{X}$ and implicitly set sk^* as the response of Q_k.

Game 9. This game is identical to Game 8, except that to answer the H_1 query. Suppose $Q_k = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$, then the system responds:

- Case 1: If $Q_k \in H_1$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ such that $T_1 = \mathsf{PK}_1, T_2 = \mathsf{PK}_2$ and $T_3 = \mathsf{shk}$, then the system responds with T_4(same as in Game 8).
- Case 2: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 1$, then the system would test the validity of z *only* using P and P^{-1} table. Formally,
  1. If $\mathsf{PK}_1 > \mathsf{PK}_2$, then the system responds with a random string;
  2. If $\mathsf{PK}_1 \leq \mathsf{PK}_2$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P^{-1}$ such that $T_4 = \mathsf{PK}_1$ and $T_1 \neq *$, then the system searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4) \in P \cup P^{-1}$ which satisfies $T'_4 = \mathsf{PK}_2$ and $T'_2 \neq *$. If such a tuple T' is found, then tests if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the system makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with a random string;
  3. If $\mathsf{PK}_1 \leq \mathsf{PK}_2$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ such that $T_4 = \mathsf{PK}_2$ and $T_1 \neq *$, then searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4)$ in P-table which satisfies $T'_4 = \mathsf{PK}_1$ and $T'_2 \neq *$. If such a tuple T' is found, then tests if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the system makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with a random string;
  4. Otherwise, responds with a random string.
- Case 3: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 0$, which means, there exists a query $Q_i = (\mathsf{SK}; H_0)(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}) \in \{\mathsf{PK}_1, \mathsf{PK}_2\}$, then the system would test the validity of z *only* using H_0-table, P-table and P^{-1}-table. Formally,
  1. There exist two queries $Q_i = ((\mathsf{SK}_1; H_0))$ and $Q_j = (\mathsf{SK}_2; H_0)(i, j < k)$ such that $\mathsf{NKG}(\mathsf{SK}_\ell) = \mathsf{PK}_\ell (\ell = 1, 2)$, then the system tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(\mathsf{keygen}(H_0(\mathsf{SK}_1)), H_0(\mathsf{SK}_2))$(the value $H_0(\mathsf{SK}_\ell)$ is extracted from the H_0-table, rather than making oracle queries). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$, otherwise responds with a random string.

2. There exists a query $Q_i = ((\mathsf{SK}^*; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}^*) = \mathsf{PK}_1$ but no such a query for PK_2, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_2$. If such a query is found, then the system tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}^*))(H_0(\mathsf{SK}^*$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}^*; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}^*)$, otherwise responds with a random string.
3. There exists a query $Q_i = ((\mathsf{SK}^*; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}^*) = \mathsf{PK}_2$ but no such a query for PK_1, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_1$. If such a query is found, then the system tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}^*))(H_0(\mathsf{SK}^*$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_1, \mathsf{SK}^*; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*)$, otherwise responds with a random string.
4. Otherwise, responds with a random string.

The difference of Game 9 and Game 8 occurs in the cases associated with underline. Specifically, instead of calling the real oracle $\tilde{H}_1$, we now lazily sample the oracle using the table for H_1. However, in both games, queries to NSK will still make queries to the random oracle $\tilde{H}_1$. As such, we will need to make sure that the response would not violate the consistency with NSK.

Note that, in either Game 9 or Game 8, the system never uses the real oracles $\tilde{H}_0, \tilde{P}$ and $\tilde{P^{-1}}$ to respond to the adversarial queries. Suppose $Q_k = (\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}^*; H_1)$, it's trivial to note that, except for negligible probability, $\mathsf{shk}^* \neq \mathsf{sharedkey}(\tilde{H}_0(\mathsf{SK}_1^*), \tilde{P}(\mathsf{PK}_2^*))$, which means that with high probability there is no $(\mathsf{PK}, \mathsf{SK})$ such that $\tilde{H}_1(\mathsf{PK}_1^*, \mathsf{PK}_2^*, \mathsf{shk}) = \widetilde{\mathsf{NSK}}(\mathsf{PK}_1^*, \mathsf{SK}_2^*)$, where $\widetilde{\mathsf{NKG}}(\mathsf{SK}_2^*) = \mathsf{PK}_2^*$. Hence, it suffices to show that the response of Q_k is consistent with the previous queries. Easy to note that for case 2.1, we can replace the response with random string, as each NSK is with form of $\mathsf{PK}_1 \leq \mathsf{PK}_2$. Moreover, it's also trivial that, for the cases where the shared key test fails (case 2.2, 2.3, 3.1, 3.2, 3.3), we can replace the responses with random strings.

The rest are case 2.4 and case 3.4, and we will explain one by one. The bad case here is that, when case 2.4 or 3.4 occurs, the adversary still can hand in a valid shared key. For case 2.4, by definition, it consists of 3 sub-cases(for ease, we denote them as case A, B and C), Case A: there is a tuple $T \in P^{-1}$ such that $T_4 = \mathsf{PK}_1^*$ but there is no tuple $T' \in P \cup P^{-1}$ that $T'_4 = \mathsf{PK}_2^*$; Case B: there is a tuple $T \in P^{-1}$ such that $T_4 = \mathsf{PK}_2^*$ but there is no tuple $T' \in P \cup P^{-1}$ that $T'_4 = \mathsf{PK}_1^*$; Case C: there is no tuple $T \in P^{-1}$ such that $T_4 \in \{\mathsf{PK}_1^*, \mathsf{PK}_2^*\}$. For case A and B, we note that the adversary might know one of the secret keys(the adversary picks sk^*, calculates $\mathsf{pk}^* = \mathsf{keygen}(\mathsf{sk}^*)$ and sets PK_1^* as the response of $(\mathsf{pk}^*; P^{-1})$), but for the other one, it even doesn't know the public key(otherwise there would be a tuple in $P \cup P^{-1}$), thus the adversary is obligated to output a valid shard-key without knowing one of the public keys. As our standard model NIKE scheme achieves entropic shared keys(the variant version), this is impossible except for negligible probability. For Case C, the

adversary might know two public keys(the adversary picks $\mathsf{SK}_1^*, \mathsf{SK}_2^*$, and sets $\mathsf{PK}_\ell^* = \mathsf{NKG}(\mathsf{SK}_\ell^*)$ then it makes two P queries $(\mathsf{PK}_1^*; P)$ and $(\mathsf{PK}_2^*; P)$ and sets pk_ℓ^* as the response of the corresponding query), but it knows nothing of the two secret keys, because no tuple is founded in H_0-table or P^{-1} table. Meanwhile, the adversary can implement the interfaces into oracles and use those oracles as additional helper to predict the shared key. Fortunately, our shared key is masked by a random oracle (H_1), thus the only thing the adversary can do is *equality test*, which means the oracles adversary implements is the best helper it can count on. And due to the semi-active unpredictable shared keys property of our standard model NIKE, we have that, the adversary can not output a valid shared key except for negligible probability.

Now we turn to case 3.4, and by definition, it consists of two sub-cases(case D and E), Case D: there is a tuple $T \in H_0$ such that $T_4 = \mathsf{PK}_1$ but there is no tuple $T' \in P \cup P^{-1}$ such that $T'_4 = \mathsf{PK}_2$; Case E: there is a tuple $T \in H_0$ such that $T_4 = \mathsf{PK}_2$ but there is no tuple $T' \in P \cup P^{-1}$ such that $T'_4 = \mathsf{PK}_1$. Similar to the analysis for case A or B, the adversary here knows one of the secret keys but does not know the other public key, so due to the entropic shared keys, it can only output a valid shared key with negligible probability.

Game 10. In Game 9, the queries to the adversarial interfaces are answered by the tables which're maintained by the system and by making queries to honest interfaces. The system never makes queries directly to $\tilde{H}_0, \tilde{P}, \tilde{P}^{-1}, \tilde{H}_1$; these oracles are *only* used to answer the $\mathsf{NKG}, \mathsf{NSK}$ queries (either generated by the adversary or by the system's response to H_0, H_1, P, P^{-1} queries). At this point, it is straightforward to show that we can replace $\mathsf{NKG}, \mathsf{NSK}$ with the ideal versions from Definition 4, resulting in Game 10.

We note that in Game 10, the system is efficient, and it responds to the adversarial interfaces just by keeping several tables and calling the honest interfaces. Thus, we can build a simulator that responds to the honest and adversarial queries precisely as the system does in Game 10. 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. As the rigorous proof is quite long, we will give the full description in Appendix E.

Remark: The rigorous proof is much more involved than the intuitive paragraphs listed above. In fact, the view of the adversary consists of the view on both the honest interfaces ($\mathsf{NKG}, \mathsf{NSK}$) and the adversarial interfaces (H_0, P, P^{-1}, H_1), and the view on those interfaces are correlated(for instance, a change on the response of a P query might affect the view on H_1). And a careful reader would have already noticed that our intuitive paragraphs only give evidence that changing the way of answering such queries would not affect the view on the corresponding interface, for instance, in Game 5, the explanation paragraph only shows the view on P would not change with high probability, which actually is insufficient, because the response of H_1 queries would be also affected in the following queries for consistency, which might change the view on H_1. Hence, in our proof, we will show that the view on *all* interfaces preserves with high probability between any two adjacent games.

References

1. E. Andreeva, A. Bogdanov, Y. Dodis, B. Mennink, and J. P. Steinberger. On the indifferentiability of key-alternating ciphers. In *Advances in Cryptology–CRYPTO 2013*, pages 531–550. Springer, 2013.
2. M. Barbosa and P. Farshim. Indifferentiable Authenticated Encryption. In *Annual International Cryptology Conference*, pages 187–220. Springer, 2018.
3. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and Efficiently Searchable Encryption. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 535–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.
5. J. Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *International Workshop on Fast Software Encryption*, pages 328–340. Springer, 2006.
6. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard pkcs #1. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 1–12, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
7. D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, Sep 2004.
8. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.
9. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. *J. ACM*, 51(4):557–594, July 2004.
10. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In *Annual International Cryptology Conference*, pages 430–448. Springer, 2005.
11. J.-S. Coron, Y. Dodis, A. Mandal, and Y. Seurin. A domain extender for the ideal cipher. In *Theory of Cryptography Conference*, pages 273–289. Springer, 2010.
12. J.-S. Coron, T. Holenstein, R. Künzler, J. Patarin, Y. Seurin, and S. Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of cryptology*, 29(1):61–114, 2016.
13. W. Diffie and M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.
14. Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 371–388. Springer, 2009.
15. Y. Dodis, M. Stam, J. Steinberger, and T. Liu. Indifferentiability of confusion-diffusion networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 679–704. Springer, 2016.
16. E. S. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In *Public-Key Cryptography–PKC 2013*, pages 254–271. Springer, 2013.
17. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, 26(1):80–101, Jan 2013.
18. S. D. Galbraith, C. Petit, B. Shani, and Y. B. Ti. On the security of supersingular isogeny cryptosystems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 63–91. Springer, 2016.

19. R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-way Permutations. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 8–26, New York, NY, 1990. Springer New York.
20. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.
21. D. Jost and U. Maurer. Security Definitions for Hash Functions: Combining UCE and Indifferentiability. In D. Catalano and R. De Prisco, editors, *Security and Cryptography for Networks*, pages 83–101, Cham, 2018. Springer International Publishing.
22. U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of cryptography conference*, pages 21–39. Springer, 2004.
23. A. Mittelbach. Salvaging Indifferentiability in a Multi-stage Setting. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 603–621, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
24. T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 487–506. Springer, 2011.
25. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
26. P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 373–390. Springer, 2006.
27. C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.
28. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
29. M. Tibouchi. Elligator Squared: Uniform Points on Elliptic Curves of Prime Order as Uniform Random Strings. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 139–156, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
30. B. Zhu, K. Li, and H. Patterson. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, pages 18:1–18:14, Berkeley, CA, USA, 2008. USENIX Association.

A Games and Ideal objects

GAMES. An n-adversary game $\mathcal{G}$ is a Turing machine, denoted as $\mathcal{G}^{\Sigma,\mathcal{A}_1,\ldots,\mathcal{A}_n}$, where Σ is a system and $\mathcal{A}_i$ are adversarial algorithms that can keep full local state but might only communicate with each other through $\mathcal{G}$. If we say a n-adversary game $\mathcal{G}_n$ is reducible to an m-adversary game $\mathcal{G}_m$, we mean that, for any $(\mathcal{A}_1,\ldots,\mathcal{A}_n)$, there are $(\mathcal{A}'_1,\ldots,\mathcal{A}'_m)$ such that for any system Σ we have that $\mathcal{G}_n^{\Sigma,\mathcal{A}_1,\ldots,\mathcal{A}_n} = \mathcal{G}_m^{\Sigma,\mathcal{A}'_1,\ldots,\mathcal{A}'_m}$. A game $\mathcal{G}$ is called a n-stage game [24] if $\mathcal{G}$ is an n-adversary game and it cannot be reducible to any m-adversary game, where $m < n$[10]. Moreover, we say two games are equivalent if they are reducible in both directions.

RANDOM ORACLE MODEL (ROM) [4]. Random oracle model is an idealized model(a theoretical black box) which responds to any unique query with a truly random string, and if the query is repeated, the response would be consistent. More concretely, a random oracle model has a publicly accessible hash function $H : \{0,1\}^* \to \{0,1\}^n$ such that : 1) for any x, every bit of $H(x)$ is truly random; 2) for any $x \neq y$, $H(x)$ and $H(y)$ are independent.

IDEAL CIPHER MODEL (ICM) [5]. The Ideal Cipher Model is another idealized model which also responds to any unique query with a truly random string. While, instead of having a publicly accessible random function, ideal cipher model has a publicly accessible ideal cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Specifically, E is an ideal cipher along with a k-bit key and n-bit input/output such that: 1) for any pair (k,x), every bit of $E(k,x)$ is truly random; 2) for any fix key k, $E(k,*)$ is a random permutation; 3) for any $k_1 \neq k_2$ and (x,y), $E(k_1,x)$ and $E(k_2,y)$ are independent. Moreover, any adversary interacting with an ideal cipher model would be given access to both the cipher and its inverse.

RANDOM FUNCTIONS. Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets, we denote $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$ to be the set of all functions that map from $\mathcal{X}$ to $\mathcal{Y}$. If we say a function $F : \mathcal{X} \to \mathcal{Y}$ is a random function, we mean that F is uniformly sampled from $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$[11].

RANDOM INJECTIONS. Similarly, let $\mathcal{X}$ and $\mathcal{Y}$ be two sets such that $|\mathcal{X}| \leq |\mathcal{Y}|$, we define $\mathcal{I}[\mathcal{X} \to \mathcal{Y}]$ to be the set of all injections that map from $\mathcal{X}$ to $\mathcal{Y}$. If we say a function $F : \mathcal{X} \to \mathcal{Y}$ is a random injection, we mean that F is uniformly sampled from $\mathcal{I}[\mathcal{X} \to \mathcal{Y}]$.

LAZY SAMPLERS. Lazy samplers are algorithmic procedures, to simulate various ideal objects along with arbitrary domain and range, by lazily sampling function at each point. Those ideal objects include: random oracle model, ideal cipher model, random functions and random injections [26].

---

[10] For any single-stage game $\mathcal{G}_{\Sigma,\mathcal{A}}$, we can trivially rewrite it as $\bar{\mathcal{A}}^{\bar{\mathcal{G}}^{\Sigma}}$, where $\bar{\mathcal{G}}$ is an oracle machine and $\bar{\mathcal{A}}$ is an adversarial algorithm, and $\bar{\mathcal{A}}$ is compatible with this oracle machine $\bar{\mathcal{G}}$.

[11] If F grants oracle accesses to all parties(honest or adversarial) in a black-box manner, then we treat F as an idealized model

B Indifferentiable Public Key Encryption

In this section, we propose the notion of "ideal PKE" and then build an indifferentiable public key encryption scheme from ideal NIKE and random oracles. Roughly, our strategy consists of two steps: first, we construct an indifferentiable deterministic public key encryption(DPKE) from an ideal NIKE, and then build an indifferentiable PKE from an ideal DPKE.

B.1 What is Ideal PKE?

In this part, we give the rigorous description of ideal PKE, formally:

Definition 6. (**Ideal PKE.**) *Let $\mathcal{X}, \mathcal{Y}, \mathcal{M}, \mathcal{R}, \mathcal{C}$ be five sets such that:* 1) $|\mathcal{X}| \geq 2^{\omega(\log \lambda)}, |\mathcal{Y}| \geq 2^{\omega(\log \lambda)}, |\mathcal{R}| \geq 2^{\omega(\log \lambda)}$ *and* $|\mathcal{C}| \geq 2^{\omega(\log \lambda)}$*;* 2) $|\mathcal{X}| \leq |\mathcal{Y}|$*;* 3) $|\mathcal{Y}| \times |\mathcal{M}| \times |\mathcal{R}| \leq |\mathcal{C}|$*. We denote $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$ as the set of all injections that map $\mathcal{X}$ to $\mathcal{Y}$; $\mathcal{E}[\mathcal{Y} \times \mathcal{M} \times \mathcal{R} \to \mathcal{C}]$ as the set of all injections that map $\mathcal{Y} \times \mathcal{M} \times \mathcal{R}$ to $\mathcal{C}$ and $\mathcal{D}[\mathcal{C} \times \mathcal{X} \to \mathcal{M} \cup \perp]$ as the set of all functions that map $\mathcal{X} \times \mathcal{C}$ to $\mathcal{M} \cup \perp$. We define $\mathcal{T}$ as the set of all function tuples (F, E, D) such that:*

- *$F \in \mathcal{F}, E \in \mathcal{E}$ and $D \in \mathcal{D}$;*
- *$\forall x \in \mathcal{X}, m \in \mathcal{M}$ and $r \in \mathcal{R}$, $D(x, E(F(x), m, r)) = m$;*
- *$\forall x \in \mathcal{X}, c \in \mathcal{C}$, if there is no $(m, r) \in \mathcal{M} \times \mathcal{R}$ such that $E(F(x), m, r) = c$, then $D(x, c) = \perp$.*

We say that a PKE scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KEYGEN}, \mathsf{PKE.ENC}, \mathsf{PKE.DEC})$, associated with secret key space $\mathcal{X}$, public space $\mathcal{Y}$, message space $\mathcal{M}$, nonce space space $\mathcal{R}$, and ciphertext space $\mathcal{C}$, is an ideal PKE if Π_{PKE} is sampled from $\mathcal{T}$ uniformly. Moreover, if the nonce space is empty, then we say such a scheme is an ideal DPKE.

Same as in Section 3, we note that, due to an information-theoretic argument, an ideal PKE achieves CCA-2 security, related-key attack security, leakage-resiliency and so forth. Next, we show how to construct an indifferentiable PKE scheme, from an ideal NIKE and random oracles.

B.2 Construction for Deterministic PKE

In this section, we build an indifferentiable *deterministic* PKE (DPKE) from simpler ideal primitives (namely random oracles and ideal ciphers) along with an ideal NIKE. We firstly present our first attempt of the construction and then illustrate a differentiator to break it (this attack also indicates a difficulty of building indifferentiable PKE). Next, we give our solution to get rid of the attack and establish the proof.

First attempt to build an indifferentiable DPKE. Given an ideal NIKE Π_{NIKE}, a natural way to build an indifferentiable DPKE is the follwing: 1) convert this ideal NIKE into a PKE scheme; 2) apply the Fujisaki-Okamoto transformation [17], which combines with a random oracle to give at least CCA-2

security. The hope is that this transformation would give us an indiffernetiable DPKE. Specifically, let $\Pi_{\mathsf{NIKE}} = (\mathsf{NKG}, \mathsf{NSK})$ be an ideal NIKE, associated with secret key space $\mathcal{X}$, public key space $\mathcal{Y}$ and shared key space $\mathcal{Z}$, and we denote $\mathsf{sk}, \mathsf{pk}, \mathsf{shk}$ to be the secret key, public key and shared key of Π_{NIKE}, respectively. For an easy exposition, we always denote the inputs of component primitives(for instance, in the standard model NIKE in Section 3 or the ideal NIKE in this section) as the lower-case and inputs of the target primitives as the upper-case. Let $\mathsf{H}_0, \mathsf{H}_1 := \{0,1\}^* \to \mathcal{X}; \mathsf{H}_2 := \{0,1\}^* \to \mathcal{Z}; \mathsf{H}_3 := \{0,1\}^* \to \mathcal{M}$, then applying FO-transform, we have the following DPKE scheme: $\Pi_{\mathsf{DPKE}} = (\mathsf{DPKE.KEYGEN}, \mathsf{DPKE.ENC}, \mathsf{DPKE.DEC})$.

- $\mathsf{DPKE.KEYGEN}(\mathsf{SK})$: On inputs secret key SK, the algorithm outputs public key $\mathsf{PK} = \mathsf{NKG}(H_0(\mathsf{SK}))$;
- $\mathsf{DPKE.ENC}(\mathsf{PK}, \mathsf{M})$: On inputs public key PK and message M, the algorithm computes $\delta = H_2(\mathsf{PK}||\mathsf{M})$, and outputs ciphertext C as

  $$\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3) = (\mathsf{NKG}(H_1(\mathsf{PK}||\mathsf{M})), \delta \oplus \mathsf{NSK}(\mathsf{PK}, H_1(\mathsf{PK}||\mathsf{M})), H_3(\mathsf{PK}, \delta) \oplus \mathsf{M});$$

- $\mathsf{DPKE.DEC}(\mathsf{SK}, \mathsf{C})$: On inputs secret key SK and ciphertext $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3)$, the algorithm computes:

  $$\mathsf{PK} = \mathsf{DPKE.KEYGEN}(\mathsf{SK}); \mathsf{A}_1 = \mathsf{NSK}(\mathsf{C}_1, H_0(\mathsf{SK}));$$
  $$\mathsf{A}_2 = \mathsf{C}_2 \oplus \mathsf{A}_1; \mathsf{A}_3 = \mathsf{C}_3 \oplus H_3(\mathsf{PK}||\mathsf{A}_2).$$

  Then it tests whether $\mathsf{C} \stackrel{?}{=} \mathsf{DPKE.ENC}(\mathsf{PK}||\mathsf{M})$. If yes, then outputs A_3, else aborts.

Correctness easily follows, but this scheme is not indifferentiable. Next we present a differetiator to break it.

Differentiator for FO transform. Due to definition, $\mathcal{D}$ has three honest interfaces $(\mathsf{DPKE.KEYGEN}, \mathsf{DPKE.ENC}, \mathsf{DPKE.DEC})$(below, we will denote $(\mathsf{DKG}, \mathsf{DE}, \mathsf{DD})$ for short) and six adversarial interfaces $(H_0, H_1, H_2, H_3, \mathsf{NKG}, \mathsf{NSK})$, and we build $\mathcal{D}$ as in Figure 3:

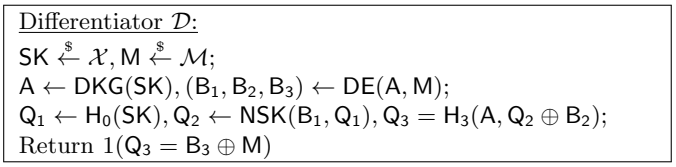

Fig. 3: Differentiator for FO-transform.

We immediately observe that, in the real game, A and $(\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3)$ are the corresponding public key and ciphertext, respectively. Moreover, due to Π_{NIKE}'s correctness, we have

$$\mathsf{NSK}(\mathsf{PK}, \mathsf{H}_1(\mathsf{PK}||\mathsf{M})) = \mathsf{NSK}(\mathsf{C}_1, \mathsf{H}_0(\mathsf{sk})) = \mathsf{NSK}(\mathsf{B}_1, \mathsf{Q}_1).$$

which refers to,

$$\Pr[\mathsf{Q}_3 = \mathsf{H}_3(\mathsf{PK}, \delta) = \mathsf{B}_3 \oplus \mathsf{M}] = 1.$$

However, in the ideal game, the simulator knows nothing of queries to the honest interfaces, and the only information it has is: $(\mathsf{SK}, \mathsf{A}, \mathsf{B}_1, \mathsf{B}_2)$(other information such like $\mathsf{H}_0(\mathsf{SK}), \mathsf{NSK}(\mathsf{B}_1, \mathsf{Q}_1)$ and $\mathsf{H}_3(\mathsf{A}, \mathsf{Q}_2 \oplus \mathsf{B}_2)$ are simulated by $\mathcal{S}$ itself). Therefore, without decryption oracle, M is independent of simulator's view. And the decryption oracle always aborts except $\mathcal{S}$ hands in a valid ciphertext, which consists of three elements $(\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3)$. Moreover, in the ideal world, the honest interface $\mathsf{DPKE.ENC}$ is a random injection, hence

$$\Pr[\mathcal{S} \text{ outputs a valid } \mathsf{B}_3] \leq \frac{\mathsf{poly}(\lambda)}{|\mathcal{M}|},$$

which means, in the ideal game,

$$\Pr[\mathsf{Q}_3 = \mathsf{H}_3(\mathsf{A}, \mathsf{Q}_2 \oplus \mathsf{B}_2) = \mathsf{B}_3 \oplus \mathsf{M}] \leq \frac{\mathsf{poly}(\lambda)}{|\mathcal{M}|}.$$

Difficulty of building indifferentiable PKE. Due to this differentiator, we know that: if we want to build an indifferentiable PKE, we should be super-cautious if the ciphertext includes piece with form of $\mathsf{string} \oplus \mathsf{M}$ [12]. And interestingly, most of the known PKE schemes have this kind of piece in ciphertext. Next, we give our solution to get rid of this kind of attack.

Our solution. To prevent the attack above, the only hope is $\mathcal{S}$ can always output a valid ciphertext itself. Our trick is, instead of using random oracles, we use an ideal cipher model P, with inverse. Specifically, let $\mathcal{Z} = \mathcal{Y} \times \mathcal{M}$ (we specify the shared key space of Π_{NIKE} to be $\mathcal{Y} \times \mathcal{M}$ and $|\mathcal{X}| \leq |\mathcal{M}|$), and $P := \mathcal{Z} \to \mathcal{Z}$. Now, we denote $\delta = P(\mathsf{PK}||\mathsf{M})$, and modify the ciphertext as $C = (\mathsf{NKG}(\mathsf{H}_1(\mathsf{PK}||\mathsf{M})), \delta \oplus \mathsf{NSK}(\mathsf{PK}, \mathsf{H}_1(\mathsf{PK}||\mathsf{M}))))$. Formally:

- $\mathsf{DPKE.KEYGEN}(\mathsf{SK})$: On inputs secret key SK, the algorithm outputs public key $\mathsf{PK} = \mathsf{NKG}(H_0(\mathsf{SK}))$;
- $\mathsf{DPKE.ENC}(\mathsf{PK}, \mathsf{M})$: On inputs public key PK and message M, the algorithm computes $\delta = P(\mathsf{PK}||\mathsf{M})$, and outputs ciphertext C as

$$\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2) = (\mathsf{NKG}(H_1(\mathsf{PK}||\mathsf{M})), \delta \oplus \mathsf{NSK}(\mathsf{PK}, \mathsf{H}_1(\mathsf{PK}, \mathsf{M})));$$

- $\mathsf{DPKE.DEC}(\mathsf{SK}, \mathsf{C})$: On inputs secret key SK and ciphertext $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2)$, the algorithm computes:

$$\mathsf{PK} = \mathsf{DPKE.KEYGEN}(\mathsf{SK}); \mathsf{A}_1 = \mathsf{NSK}(\mathsf{C}_1, H_0(\mathsf{SK}));$$

$$\mathsf{A}_2 = \mathsf{C}_2 \oplus \mathsf{A}_1; \mathsf{A}_3 = P^{-1}(A_2), \mathsf{A}_4 = \mathsf{A}_3/\mathsf{PK}.$$

  Then it tests whether $\mathsf{C} \stackrel{?}{=} \mathsf{DPKE.ENC}(\mathsf{A}_3)$. If yes, then outputs A_4[13], else aborts.

[12] we stress that this difficulty might not be an impossibility separation, and here we just point out a vulnerability of building indifferentiable PKEs

[13] we note that $\mathsf{A}_3 = \mathsf{PK}||\mathsf{M}$, and by A_3/PK, we mean removing PK from A_3.

In our new setting, we immediately observe that the ciphertext only consists of *two* elements, which means $\mathcal{S}$ can always hand in the vaild ciphertext to the decryption oracle. Now we see how this new scheme prevent the attack above. Following the same spirit, the corresponding differentiator for the new scheme should be:

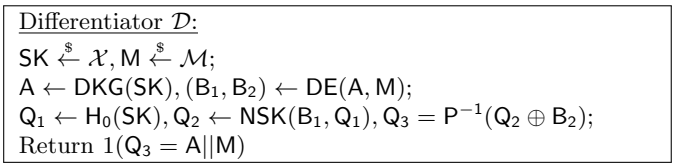

Fig. 4: Differentiator for new scheme.

Same as above, in real game, $\Pr[\mathcal{D}] = 1$. Meanwhile, in ideal game, we note that $\mathcal{S}$ still can extract the following information $(\mathsf{SK}, \mathsf{A}, \mathsf{B}_1, \mathsf{B}_2)$. As now the ciphertext only consists of two elements, $\mathcal{S}$ can run the decryption oracle $\mathsf{M}' \leftarrow \mathsf{DD}(\mathsf{SK}, (\mathsf{B}_1, \mathsf{B}_2))$ and output $\mathsf{Q}_3 = \mathsf{A}||\mathsf{M}'$. Hence, in ideal game, we also have $\Pr[\mathcal{D}] = 1$. Apparently, this is *only* an evidence that our new scheme prevents this specific differentiator. And to prove it's an indifferentiable DPKE, we need to show that our scheme can prevent all kind of PPT differetiators.

Theorem 7. (Indifferentiable DPKE). Π_{DPKE} *is indifferentiable from an ideal DPKE if* Π_{NIKE} *is an ideal NIKE.*

As the proof of Theorem 7 is long, we will give the full details in Appendix F.

B.3 Construction for PKE

In this section, we complete the construction by building an indifferentiable PKE from ideal DPKE and random oracles. Similarly as in Appendix B.2 , we firstly present two attempts and then illustrate the corresponding differentiators to break the schemes. Then we give the modified solution to get rid of those attacks and complete the proof.

First attempt. Given an ideal DPKE $\Pi_{\mathsf{DPKE}} = (\mathsf{DKG}, \mathsf{DE}, \mathsf{DD})$, a trivial way to build a PKE scheme is to treat the nonce as part of the message and discard it in the decryption procedure. Specifically:

- $\mathsf{PKE.KEYGEN}(\mathsf{SK}) = \mathsf{DKG}(\mathsf{SK})$;
- $\mathsf{PKE.ENC}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \mathsf{DE}(\mathsf{PK}, \mathsf{M}||\mathsf{R})$;
- $\mathsf{PKE.DEC}(\mathsf{SK}, \mathsf{C})$: On inputs a secret key SK and a ciphertext C, the algorithm runs $\mathsf{DD}(\mathsf{SK}, \mathsf{C})$. If aborts then it aborts, else let $(\mathsf{M}||\mathsf{R}) = \mathsf{DD}(\mathsf{SK}, \mathsf{C})$, it outputs M.

Correctness of the scheme is straightforward, while unfortunately, it's not indifferentiable.

Differentiator for first attempt. Due to definition, $\mathcal{D}$ has three honest interfaces ($\mathsf{PKE.KEYGEN}, \mathsf{PKE.ENC}, \mathsf{PKE.DEC}$)(below, we will denote ($\mathsf{PKG}, \mathsf{PE}, \mathsf{PD}$) for ease) and three adversarial interfaces ($\mathsf{DKG}, \mathsf{DE}, \mathsf{DD}$). We build $\mathcal{D}$ as in Figure 5:

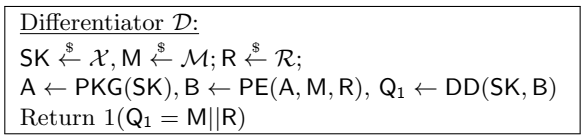

Fig. 5: Differentiator for first attempt.

Easy to note that, in real game $\Pr[\mathcal{D} = 1] = 1$. Meanwhile, in ideal game, the simulator has no information of the nonce R, as PD only outputs M. To respond to the query, the simulator has to guess the nonce itself, hence we have

$$\Pr[\mathcal{D} = 1] \leq \frac{\mathsf{poly}(\lambda)}{|\mathcal{R}|}.$$

Second attempt. To solve the problem above, we add an additional random oracle to force the attacker to hand in the nonce when calling the DE-query. Specifically, let $H_0 := \{0,1\}^* \to \mathcal{R}$, then we build Π_{PKE} as:

- $\mathsf{PKE.KEYGEN}(\mathsf{SK}) = \mathsf{DKG}(\mathsf{SK})$;
- $\mathsf{PKE.ENC}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \mathsf{DE}(\mathsf{PK}, \mathsf{M}||H_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}))$;
- $\mathsf{PKE.DEC}(\mathsf{SK}, \mathsf{C})$: On inputs a secret key SK and a ciphertext C, the algorithm runs $\mathsf{DD}(\mathsf{SK}, \mathsf{C})$. If DD aborts then the algorithm aborts, else let $(\mathsf{M}||\mathsf{str}) = \mathsf{DD}(\mathsf{C}, \mathsf{SK})$, it outputs M.

Apparently, this modification can prevent the attack above, as the response of the DD query now is $H_0(\mathsf{PK}, \mathsf{M}||\mathsf{R})$ and H_0 is under $\mathcal{S}$'s control (To test the consistency, the adversary has to hand in R to $\mathcal{S}$, as a result the simulator can always output the proper response). Although the attack above is solved, unfortunately, this scheme is still insufficient.

Differentiator for second attempt. Due to definition, any differentiator has three honest interfaces ($\mathsf{PKG}, \mathsf{PE}, \mathsf{PD}$)and four adversarial interfaces ($\mathsf{DKG}, \mathsf{DE}$,$\mathsf{DD}$, H_0). We build $\mathcal{D}$ as in Figure 6:

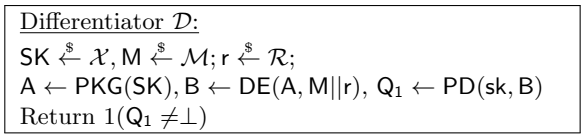

Fig. 6: Differentiator for second attempt.

Easy to note that, in real game, $\Pr[\mathcal{D} = 1] = 1$. Meanwhile, in ideal game, we claim that, with noticeable probability, the decryption would abort. In fact,

the decryption procedure outputs M if and only if there exists a nonce $\mathsf{R} \in \mathcal{R}$ such that $H_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \mathsf{r}$. Moreover, $\mathsf{R}, \mathsf{r} \in \mathcal{R}$, and H_0 is a random oracle, we have that

$$\Pr[\forall \mathsf{R} \in \mathcal{R}, H_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}) \neq r] \approx 1/e$$

which refers to $\Pr[\mathcal{D} = 1] \leq 1 - 1/e \approx 0.6$

Our solution. To get rid of this new attack, we have to shorten the size of H_0's range, to make sure that every element in H_0's range has pre-image with high probability. Meanwhlie, to make sure the ciphertext space is sufficiently large, we also need to pad some dummy strings. Specifically, let Π_{DPKE} be an ideal DPKE, associated with secret key space $\mathcal{X}$, public key space $\mathcal{Y} = \{0,1\}^{n_1}$, message space $\mathcal{M} = \{0,1\}^{n_2+2n_3}$, and ciphertext space $\mathcal{C}$, and let $H_0 := \{0,1\}^* \to \{0,1\}^{n_3}$, where $n_2 > 0$ and $n_1, n_3 \geq \omega(\log \lambda)$, then we build our scheme as:

- $\mathsf{PKE.KEYGEN}(\mathsf{SK}) = \mathsf{DKG}(\mathsf{SK})$:
- $\mathsf{PKE.ENC}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$: On inputs public key PK, message $\mathsf{M} \in \{0,1\}^{n_2}$ and nonce $\mathsf{R} \in \{0,1\}^{2n_3}$, the algorithm outputs ciphertext:

$$\mathsf{C} = \mathsf{DE}(\mathsf{PK}, \mathsf{M} || H_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}) || \underbrace{0 \ldots 0}_{n_3});$$

- $\mathsf{PKE.DEC}(\mathsf{SK}, \mathsf{C})$: On inputs secret key SK and ciphertext C, the algorithm runs $\mathsf{A} = \mathsf{D}_{\mathsf{DPKE}}(\mathsf{SK}, \mathsf{C})$. If $\mathsf{A} = \perp$ or the last n_3 bits are not 0^{n_3}, then it aborts, else the algorithm outputs the first n_2 bits.

Correctness follows easily, and the rest is to show its indifferentiability.

Theorem 8. (Indifferentiable PKE). Π_{PKE} *is indifferentiable from an ideal PKE if* Π_{DPKE} *is an ideal DPKE.*

Again, as the proof of Theorem 8 is quite long, we put the full details into Appendix G.

C Indifferentiable Digital Signatures

In this section, we extend our result to the digital signature scheme. We propose the notion of "Ideal Signature", and then build an indifferentiable signature scheme from simpler ideal primitives (random oracle model and ideal cipher model) and a stand-model signature scheme.

C.1 What is "Ideal Signature"?

In this part, we give the rigorous description of ideal signature, formally:

Definition 9. (**Ideal Signature.**) *Let* $\mathcal{X}, \mathcal{Y}, \mathcal{M}, \Sigma$ *be four sets such that:* 1) $|\mathcal{X}| \geq 2^{\omega(\log \lambda)}, |\mathcal{Y}| \geq 2^{\omega(\log \lambda)}, |\mathcal{M}| \geq 2^{\omega(\log \lambda)}$ *and* $|\Sigma| \geq 2^{\omega(\log \lambda)}$*;* 2) $|\mathcal{X}| \leq |\mathcal{Y}|$*;* 3) $|\mathcal{X}| \times |\mathcal{M}| \leq |\Sigma|$*. We denote* $\mathcal{F}[\mathcal{X} \to \mathcal{Y}]$ *as the set of all injections that map* $\mathcal{X}$ *to* $\mathcal{Y}$*;* $\mathcal{S}[\mathcal{X} \times \mathcal{M} \to \Sigma]$ *as the set of all injections that map* $\mathcal{X} \times \mathcal{M}$ *to* Σ *and* $\mathcal{V}[\mathcal{Y} \times \mathcal{M} \times \Sigma \to \{0,1\}]$ *as the set of all functions that map* $\mathcal{Y} \times \mathcal{M} \times \Sigma$ *to a bit. We define* $\mathcal{T}$ *as the set of all function tuples* (F, S, V) *such that:*

- $F \in \mathcal{F}, S \in \mathcal{S}$ *and* $V \in \mathcal{V}$*;*
- $\forall x \in \mathcal{X}, m \in \mathcal{M}$*,* $V(F(x), m, S(x, m)) = 1$*;*
- $\forall x \in \mathcal{X}, m \in \mathcal{M}$ *and* $\sigma \in \Sigma$*, if* $\sigma \neq \mathsf{sign}(x, m)$*, then* $V(F(x), m, \sigma) = 0$*;*
- $\forall x \in \mathcal{X}, m \in \mathcal{M}$ *and* $\sigma_1, \sigma_2 \in \Sigma$*,* $V(F(x), m, \sigma_1) = V(F(x), m, \sigma_2) = 1 \Rightarrow \sigma_1 = \sigma_2$*.*

We say that a digital signature scheme $\Pi_{\mathsf{Sig}} = (\mathsf{Sig.KEYGEN}, \mathsf{Sig.SIGN}, \mathsf{Sig.VER})$*, associated with secret key space* $\mathcal{X}$*, public key space* $\mathcal{Y}$*, message space* $\mathcal{M}$ *and signature space* Σ*, is an ideal digital signature, if* Π_{Sig} *is sampled from* $\mathcal{T}$ *uniformly.*

It's trivial to note that, an ideal signature is a unique signature, and due to the information-theoretic argument, it also achieves CMU-secure, related-key attack secure, leakage-resilient and so forth. Next, we show how to construct an indifferentiable signature scheme from simpler ideal primitives.

C.2 Construction

In this section, we build our indifferentiable signature scheme from simpler ideal primitives (namely random oracles and ideal ciphers) along with a standard-model(that is, *non-ideal*) signature scheme.

Building Blocks. Our scheme will consist of several building blocks:

- A standard-model signature scheme $\Pi_{\mathsf{SM-Sig}} = (\mathsf{keygen}, \mathsf{sign}, \mathsf{ver})$ with secret key space $\mathcal{X}$, public key space $\mathcal{Y} = \{0,1\}^{n_1}$, message space $\mathcal{M} = \{0,1\}^{n_2}$ and signature space $\Sigma \subset \{0,1\}^{n_3}$;
- $H_0 := \{0,1\}^* \to \mathcal{X}; H_1 := \{0,1\}^* \to \mathcal{M}$;
- $P := \{0,1\}^{n_1} \to \{0,1\}^{n_1}$ is a random permutation and P^{-1} is P's inverse,
- $E := \{0,1\}^{n_1+n_2} \times \{0,1\}^{n_3} \to \{0,1\}^{n_3}$ is an ideal cipher model, where $\{0,1\}^{n_1+n_2}$ is its key space and E^{-1} is its inverse.

Construction. Now we are ready to build an indifferentiable signature scheme, denoted as $\Pi_{\mathsf{Sig}} = (\mathsf{Sig.KEYGEN}, \mathsf{Sig.SIGN}, \mathsf{Sig.VER})$, from the building blocks above. Formally,

- $\mathsf{Sig.KEYGEN}(\mathsf{SK})$: On inputs secret key SK, the algorithm outputs public key $\mathsf{PK} = P^{-1}(\mathsf{keygen}(H_0(\mathsf{SK})))$;
- $\mathsf{Sig.SIGN}(\mathsf{SK}, \mathsf{M})$: On inputs secret key SK and message M, the algorithm computes $\mathsf{PK} = \mathsf{Sig.KEYGEN}(\mathsf{SK})$ and outputs the signature

$$\sigma = E^{-1}((\mathsf{PK}||\mathsf{M}), \mathsf{sign}(H_0(\mathsf{SK}), H_1(\mathsf{M}))),$$

- $\mathsf{Sig.VER}(\mathsf{PK}, \mathsf{M}, \sigma)$: On inputs public key PK, message M and the signature σ, the algorithm outputs a bit $b = \mathsf{ver}(P(\mathsf{PK}), H_1(\mathsf{M}), E((\mathsf{PK}||\mathsf{M}), \sigma))$.

Correctness easily follows, and the rest is to prove its indifferentiability. Before that, we specify several security properties of the standard-model signature.

Property 1. UNIQUENESS. We say a signature achieves uniqueness, if $\forall(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{keygen}, \mathsf{m} \in \mathcal{M}, \sigma_1, \sigma_2 \in \Sigma$, we have,

$$\mathsf{ver}(\mathsf{pk},\mathsf{m},\sigma_1) = \mathsf{ver}(\mathsf{pk},\mathsf{m},\sigma_2) = 1 \Rightarrow \sigma_1 = \sigma_2.$$

Property 2. RANDOM-MESSAGE ATTACK (RMA). We say a signature scheme is RMA-secure if for any PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}_{\mathcal{A}} := \Pr[\mathsf{ver}(\mathsf{pk},\mathsf{m}^*,\sigma^*) = 1 : \sigma^* \leftarrow \mathcal{A}^{\mathsf{sign}(\mathsf{sk},\mathsf{m}_1,\ldots,\mathsf{m}_\mathsf{q})}(\mathsf{pk},\mathsf{m}^*)] \leq \mathsf{negl}(\lambda)$$

where $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{keygen}, (\mathsf{m}^*,\mathsf{m}_1,\ldots,\mathsf{m}_\mathsf{q}) \xleftarrow{\$} \mathcal{M}$ and m^* was not previously signed.

Property 3. PSEUDORANDOM PUBLIC KEY. We say the public key, for a signature scheme, is pseudorandom, if for any PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}_{\mathcal{A}} := |\Pr[\mathcal{A}(\mathsf{keygen}(r))] - \Pr[\mathcal{A}(R)]| \leq \mathsf{negl}(\lambda)$$

where $r \leftarrow \mathcal{X}, R \leftarrow \mathcal{Y}$.

We say a signature scheme is Good if it satisfies the three properties above. Next, we prove the following theorem.

Theorem 10. (Indifferentiable Signatures). Π_{Sig} *is indifferentiable from an ideal digital signature if* $\Pi_{\mathsf{SM-Sig}}$ *is* Good.

Again, as the proof of Theorem 10 is quite long, we put the full details into Appendix H.

D Instantiating Our Constructions

In this section, we explain how to instantiate our schemes. In particular, we briefly explain how to build secure standard-model schemes that can be plugged into our construction.

D.1 Non-interactive Key Exchange (NIKE)

We need a NIKE protocol where (1) the shared key is semi-active computationally unpredictable; (2) for any secret key, the distribution of shared key is close to uniform in shared key space over the public key's distribution and (3) the public values exchanged by the parties are computationally close to uniform bit strings. (1) is a weakening of the usual notion of security for a NIKE scheme against active attack, and (2) is guaranteed in most known schemes, except for some artificial constructions. In contrast, (3) is non-standard, and requires some care to make sure it is handled properly. We now give several example constructions:

From doubly-strong CDH. The usual Diffie-Hellman protocol gives a NIKE that is almost good enough for us. Let G be a group of prime order p with generator g. The secret keys are integers $a \in \mathbb{Z}_p$. $\mathsf{keygen}(a) = g^a$ and $\mathsf{sharedkey}(h, a) = h^a$. For $g^a = \mathsf{keygen}(a)$ and $g^b = \mathsf{keygen}(b)$, the secret shared key is $\mathsf{sharedkey}(g^a, b) = g^{ab}$. The doubly-strong computational Diffie-Hellman assumption states that it is computationally infeasible to compute g^{ab} from g^a, g^b and having access to an oracle that on input (u, v) tests whether $v = u^a$ and another oracle that on input (u, v) tests whether $v = u^b$. This assumption can plausibly hold on typical cryptography groups, and over bilinear groups, it follows trivially from the plain CDH assumption, since the bilinear map can be used to implement the oracles. Moreover, this assumption immediately gives us the semi-active unpredictable shared keys in our standard model NIKE protocol.

For the pseudorandomness of public keys, we need some more care. We consider two cases:

- G is the group of quadratic residues mod a safe prime q. In other words, $q = 2p + 1$. In this case, the public key is a random residue, clearly not a uniformly random bit string
- G is an elliptic curve over a finite field. Here, the public key is a random point on the elliptic curve, which is represented by two elements of the finite field satisfying the elliptic curve equation. Again, clearly not a uniform bit string.

We now explain how to convert either protocol into one with statistically random public keys.

In the case of the group of quadratic residues, we will assume $q = 3 \bmod 4$. In this case, -1 is *not* a residue, meaning that for every non-zero integer x in $\mathbb{Z}_q$, exactly one of x and $-x$ is a reside. As such, we can bijectively map the group G to the integers $\{1, \ldots, p\}$: for each element $g \in G$, if $g \in \{1, \ldots, p\}$ output g, and otherwise output $q - g$.

We are still not done: now the public key is random in $\{1, \ldots, p\}$, but as p is not a power of 2, this cannot be represented as a uniform bit string. Instead, we will use the following:

Lemma 11. *Let D be a distribution over $[0, N-1]$ that is computationally indistinguishable from uniform. Then n be an integer such that $n - \log_2 N \geq \lambda$ where λ is the security parameter. Let T be the largest integer such that $TN \leq 2^n$. Then the distribution $x + Ny$ where $x \leftarrow S$ and $y \leftarrow [0, T-1]$ is computationally indistinguishable from the uniform distribution over $[0, 2^n - 1]$.*

Proof. We prove this by a sequence of hybrids:

Hybrid 0. In this case, the adversary is given $z = x + Ny$ for $x \leftarrow D$ and $y \leftarrow [0, T-1]$.

Hybrid 1. In this case, the adversary is given $z = x + Ny$ for $x \leftarrow [0, N-1], y \leftarrow [0, T-1]$. Notice that Hybrids 0 and 1 are computationally indistinguishable since D is computationally indisitnguishable from uniform on $[0, N-1]$. Notice that in Hybrid 2, z is uniform on $[0, TN-1]$

Hybrid 2. In this hybrid, z is uniform on $[0, 2^n]$. Notice that $\|2^n - TN\| \leq N$. Therefore the two distributions are at most $N/2^n$-close in statistical distance. Notice that $N/2^n \leq \frac{1}{2^\lambda}$, which is negligible.

Lemma 11 shows that any key generation algorithm where outputs are uniform in some interval can be turned into an algorithm where outputs are uniform bit strings. By applying this Lemma to Diffie-Hellman over safe primes, we obtain a NIKE scheme meeting our needs.

In the case of elliptic curves, things are a bit more complicated. Fortunately, there are multiple ways to represent elliptic curve elements as (almost) random elements in the ambient finite field; for example, see [29]. Elements in the finite field of q elements can be mapped into the integers $[0, q-1]$ in a natural way, and then we can apply Lemma 11.

D.2 Signatures

Next, we turn to signatures. Unfortunately, as we need unique signatures, we have to be careful about how we instantiate our scheme.

From Bilinear maps. We can use the unique signature scheme of Boneh, Lynn, and Shacham [7] from bilinear maps. Since we need a weaker notion of security than the usual chosen message attack security, we can actually use a simplification of their scheme.

The secret key is an integer $a \in \mathbb{Z}_p$, and the public key is $h = g^a$. Messages m are just group elements, and the signature on m is $\sigma = m^a$. Notice that (g, h, m, σ) is a Diffie-Hellam tuple; therefore, all that's needed to verify the signature is a way to solve the decisional Diffie-Hellam (DDH) problem. On the other hand, by the computational Diffie-Hellman (CDH) problem it is hard to compute σ from g, h, m. Such "gap Diffie-Hellman" groups can be instantiated from bilinear maps.

The only thing that remains is to guarantee that public keys are random bit strings. Since known constructions of bilinear maps are just special elliptic curve groups, we can apply the same modifications as we did for key agreement above to get random public keys.

E Proof of Theorem 5

In this section, we give the full description of our simulator and the rest proof of Theorem 5.

Simulator In Ideal Game. Let $(\mathsf{KEYGEN}, \mathsf{SHAREDKEY})$ be the function pair that samples from $\mathcal{T}_{\mathsf{NIKE}}$, the simulator works as follows. Like the system in Game 10, the simulator also maintains four tables, referring to H_0-table, P-table, P^{-1}-table and H_1-table. Concretely:

- H_0-table: Initially empty, consists of tuples with form of $(\mathsf{SK}, \mathsf{sk}, \mathsf{pk}, \mathsf{PK})$.

- P-table: Initially empty, consists of tuples with form of $(*, \mathsf{sk}, \mathsf{pk}, \mathsf{PK})$.
- P^{-1}-table: Initially empty, consists of tuples with form of $(\mathsf{SK}, *, \mathsf{pk}, \mathsf{PK})$.
- H_1-table: Initially empty, consists of tuples with form of $(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}, \mathsf{SHK})$.

By definition, $\mathcal{S}$ has access to the honest interfaces $(\mathsf{NKG}, \mathsf{NSK})$, where $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{KEYGEN}(\mathsf{SK})$ and $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, \mathsf{SK}) = \mathsf{SHAREDKEY}(\mathsf{PK}, \mathsf{SK})$. And for the adversarial queries, $\mathcal{S}$ works as the system in Game 10, by just using the tables and calling the honest interfaces.

H_0-QUERY. Suppose $Q_k = (\mathsf{SK}; H_0)$ $(k \in [1, q])$, then the simulator responds as follows:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{SK}$, then $\mathcal{S}$ responds with T_2;
- Case 2. If $Q_k \notin H_0$ but there is a tuple $T = (T_1, T_2, T_3, T_4)$ in P-table such that $T_4 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$, then $\mathcal{S}$ responds with T_2 and inserts $(\mathsf{SK}, T_2, T_3, T_4)$ into H_0-table;
- Case 3. Otherwise, the simulator samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds with sk, and inserts $(\mathsf{SK}, \mathsf{sk}, \mathsf{keygen}(\mathsf{sk}), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0-table.

P-QUERY. Suppose $Q_k = (\mathsf{PK}; P)$, then the simulator responds:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P \cup P^{-1}$ such that $T_4 = \mathsf{PK}$, then $\mathcal{S}$ responds with T_3;
- Case 2. Otherwise, $\mathcal{S}$ responds with a random public key $\mathsf{pk}^* = \mathsf{keygen}(\mathsf{sk}^*)$, where $\mathsf{sk}^* \xleftarrow{\$} \mathcal{X}$ and inserts $(*, \mathsf{sk}^*, \mathsf{keygen}(\mathsf{sk}^*), \mathsf{PK})$ into P-table.

P^{-1}-QUERY. Suppose $Q_k = (\mathsf{pk}; P^{-1})$, then the simulator responds:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P \cup P^{-1}$ such that $T_3 = \mathsf{pk}$, then $\mathcal{S}$ responds with T_4;
- Case 2. Otherwise, $\mathcal{S}$ samples $\mathsf{SK} \leftarrow \mathcal{X}$, makes a query $(\mathsf{SK}; \mathsf{NKG})$, responds to the query with $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ and inserts $(\mathsf{SK}, *, \mathsf{pk}, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into P^{-1}-table.

H_1-QUERY. Suppose $Q_k = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$, then the simulator responds:

- Case 1: If $Q_k \in H_1$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ such that $T_1 = \mathsf{PK}_1, T_2 = \mathsf{PK}_2$ and $T_3 = \mathsf{shk}$, then $\mathcal{S}$ responds with T_4.
- Case 2: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 1$, then $\mathcal{S}$ would test the validity of z *only* using P and P^{-1} table. Formally,
  1. If $\mathsf{PK}_1 > \mathsf{PK}_2$, then $\mathcal{S}$ responds with a random string;
  2. If $\mathsf{PK}_1 \leq \mathsf{PK}_2$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P^{-1}$ such that $T_4 = \mathsf{PK}_1$, then $\mathcal{S}$ searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4) \in P \cup P^{-1}$ which satisfies $T'_4 = \mathsf{PK}_2$. If such a tuple T' is found, then test if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the simulator makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with a random string;

3. If $\mathsf{PK}_1 \leq \mathsf{PK}_2$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ such that $T_4 = \mathsf{PK}_2$, then $\mathcal{S}$ searches every tuple $T' = (T'_1, T'_2, T'_3, T'_4)$ in P-table which satisfies $T'_4 = \mathsf{PK}_1$. If such a tuple T' is found, then test if $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, T'_2)$(only using the tuples in $P \cup P^{-1}$). If so, the simulator makes a query $(T'_4, T_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(T'_4, T_1)$, otherwise responds with a random string;
4. Otherwise, responds with a random string.

– Case 3: If $Q_k \notin H_1$ and $\mathsf{WCC}_k = 0$, which means, there exists a query $Q_i = (\mathsf{SK}; H_0)(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}) \in \{\mathsf{PK}_1, \mathsf{PK}_2\}$, then $\mathcal{S}$ would test the validity of z *only* using H_0-table, P-table and P^{-1}-table. Formally,
1. There exist two queries $Q_i = ((\mathsf{SK}_1; H_0))$ and $Q_j = (\mathsf{SK}_2; H_0)(i, j < k)$ such that $\mathsf{NKG}(\mathsf{SK}_\ell) = \mathsf{PK}_\ell(\ell = 1, 2)$, then the simulator tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(\mathsf{keygen}(H_0(\mathsf{SK}_1)), H_0(\mathsf{SK}_2))$(the value $H_0(\mathsf{SK}_\ell)$ is extracted from the H_0-table, rather than making oracle queries). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}_1; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$, otherwise responds with a random string.
2. There exists a query $Q_i = ((\mathsf{SK}^*; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}^*) = \mathsf{PK}_1$ but no such a query for PK_2, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_2$. If such a query is found, then $\mathcal{S}$ tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}^*))(H_0(\mathsf{SK}^*)$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_2, \mathsf{SK}^*; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}^*)$, otherwise responds with a random string.
3. There exists a query $Q_i = ((\mathsf{SK}^*; H_0))(i < k)$ such that $\mathsf{NKG}(\mathsf{SK}^*) = \mathsf{PK}_2$ but no such a query for PK_1, then searches every tuple $T = (T_1, T_2, T_3, T_4) \in P \cup P^{-1}$ which satisfies $T_4 = \mathsf{PK}_1$. If such a query is found, then $\mathcal{S}$ tests $\mathsf{shk} \stackrel{?}{=} \mathsf{sharedkey}(T_3, H_0(\mathsf{SK}^*))(H_0(\mathsf{SK}^*)$ is extracted from H_0-table and T_3 is from the $P \cup P^{-1}$-table). If so, the system makes a query $(\mathsf{PK}_1, \mathsf{SK}^*; \mathsf{NSK})$ and responds to Q_k with $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*)$, otherwise responds with a random string.
4. Otherwise, responds with a random string.

Next we prove the indistinguishability between any adjacent games, by showing that the adversary's view on both games are close. Before that, we give two useful lemmas based on $\Pi_{\mathsf{SM-NIKE}}$'s pseudorandom public keys and entropic shared keys.

Lemma 12. *Let* $X := \mathsf{keygen}(\mathsf{sk})$ *be a variable, where* $\mathsf{sk} \leftarrow \mathcal{X}$*, then* $\forall y \in \mathcal{Y}$*,* $\Pr[X = y] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}$.

Proof. By definition, the advantage of pseudorandom public key is bounded by ϵ_3, hence by hybrid argument, we have that

$$\mathrm{Adv}_{\mathcal{A}} := |\Pr[\mathcal{A}(\mathsf{keygen}(\mathsf{sk}_1), \mathsf{keygen}(\mathsf{sk}_2))] - \Pr[\mathcal{A}(y_1, y_2)]| \leq 2\epsilon_3$$

where $\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathcal{X}, y_1, y_2 \leftarrow \mathcal{Y}$.

Assuming there exists y^* such that $\Pr[X = y^*] > \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}$, then it's trivial that $\Pr[\mathsf{keygen}(\mathsf{sk}_1) = \mathsf{keygen}(\mathsf{sk}_2)] > 2\epsilon_3 + \frac{1}{|\mathcal{X}|}$. On the other hand, $\Pr[y_1 = y_2] = \frac{1}{|\mathcal{Y}|} \leq \frac{1}{|\mathcal{X}|}$, which means we can differ the tuple with a better advantage than $2\epsilon_3$.

Similar to Lemma 12 , we also have the following lemma.

Lemma 13. *Let* sk^* *be any fixed element in* $\mathcal{X}$*,* pk^* *be any fixed element in* $\mathcal{Y}$*, and* $Z_1 = \mathsf{sharedkey}(\mathsf{pk}, \mathsf{sk}^*)$ *and* $Z_2 = \mathsf{sharedkey}(\mathsf{pk}^*, \mathsf{sk})$ *where* $\mathsf{sk} \leftarrow \mathcal{X}, \mathsf{pk} \leftarrow \mathcal{Y}$*, then* $\forall z \in \mathcal{Z}, \ell \in \{1, 2\}$, $\Pr[Z_\ell = z] \leq \epsilon_2$

Note that Lemma 12 tells us that the public key has high min-entropy, and any PPT adversary cannot predict it with a noticeable probability, without knowing the corresponding secret key. Analogously, Lemma 13 tells us that any PPT adversary cannot predict the shared key without knowing one party's public key. Now, we are ready to prove the indistinguishability between any adjacent games.

Claim. Game Real $\approx$ Game 0.

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

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

Claim. Game 0 $\approx$ Game 1.

Proof. By definition, we note that in Game 1, the system maintains longer tables and it responds to part of the queries (called type A queries) by just using those tables and calling the honest interfaces. For the rest queries (called type B queries), the system responds by calling the real oracles. Moreover, the items stored in those tables are always consistent with the real oracles, and in either games, the honest interfaces correspond to the real oracles, which means the response by calling the honest interfaces is identical to the one by calling real oracles(for instance, $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{NKG}}(\mathsf{SK}) = \tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})))$). Hence the responses of type A queries by either the real oracles(Game 0) or by honest interfaces plus tables(Game 1) are identical, which refers to

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

Claim. Game 1 $\approx$ Game 2.

Proof. Recalling that the only difference between Game 1 and Game 2 occurs in case 2, where $Q_k = (\mathsf{PK}^*; P) \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$. In Game 1, the system responds to Q_k with $\tilde{P}(\mathsf{PK}^*)$ while in Game 2, the system replaces the response with a random string in $\mathcal{Y}$. To prove the indistinguishability, we first formalize the adversary's view in Game 1. By definition, we immediately observe that, in Game 1,

- The system responds to (SK, H_0) with $\tilde{H}_0(\mathsf{SK})$;
- The system responds to $(\mathsf{PK}; P)$ with $\tilde{P}(\mathsf{PK})$;
- The system responds to $(\mathsf{pk}; P^{-1})$ with $\tilde{P}^{-1}(\mathsf{pk})$;
- The system responds to $(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$ with $\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk})$;
- The system responds to $(\mathsf{SK}; \mathsf{NKG})$ with $\widetilde{\mathsf{NKG}}(\mathsf{SK})$;
- The system responds to $(\mathsf{PK}_1, \mathsf{SK}_2, \mathsf{NSK})$ with $\widetilde{\mathsf{NSK}}(\mathsf{PK}_1, \mathsf{SK}_2)$.

Hence in adversary's perspective, under the consistency conditions listed below, the responses of H_0 and H_1 are independent and random strings. For P queries $Q_k = (\mathsf{PK}; P) \notin H_0 \cup P \cup P^{-1}$, if $\mathsf{SCC}_k = 1$, then the response is uniformly distributed in $\mathcal{Y}$, while if $\mathsf{SCC}_k = 0$, then the response should be a random public key instead. For P^{-1} query $Q = (\mathsf{pk}; P^{-1})$, the response should be the inverse of P, and if $Q \notin H_0 \cup P \cup P^{-1}$, then the response is independent and uniformly distributed. Here are the consistency conditions:

1. $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk})) = \mathsf{pk}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{PK})) = \mathsf{PK}$;
2. There exists no $(\mathsf{PK}_1 \neq \mathsf{PK}_2), (\mathsf{pk}_1 \neq \mathsf{pk}_2)$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_2)$ or $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$;
3. $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK})))$;
4. $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1^{r}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$;
5. $P^{r}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}))$(case 2 for H_0 query);
6. $H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)), \mathsf{sk}_1))$
   $= \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2)$ (case 2.2, 2.3 for H_1 query) ;
7. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{PK}_2, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_2), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$ (case 3.2, 3.3 for H_1 query);
8. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$
   $= \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{SK}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{SK}_1)$(case 3.1 for H_1 query) .

Now we turn to Game 2, where the system responds to Q_k ($Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$) with a random string y in $\mathcal{Y}$, comparing to $\tilde{P}(\mathsf{PK}^*)$ in Game 1. We note that:

- for H_0 query, the response is identical in either game, as $H_0^{\mathsf{r}}(\mathsf{SK}) = \tilde{H}_0(\mathsf{SK})$(the second term of every tuples in P tables is *).
- for P query, the response of Q_k is well-distributed as y is uniformly sampled. And after Q_k, the system implicitly sets $P^{-1\mathsf{r}}(y) = \mathsf{PK}^*$.
- for P^{-1} query, as long as y never appears before Q_k (say $\mathcal{A}$ makes a query (y, P^{-1}) before Q_k), the responses of P^{-1} are well-formed and consistent.
- for H_1 query, after Q_k the system implicitly sets

$$\begin{aligned} & H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(y, H_0^{\mathsf{r}}(\mathsf{SK}_1)) \\ =& \tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1)) \\ =& \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1) \end{aligned}$$

  where $\mathsf{PK}_1 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1)$. Hence, as long as the adversary does not make a query $(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1); H_1)$, the responses of H_1 queries are well distributed ($\tilde{H}_1$ is a random oracle).

For the bad event on P^{-1} query(Bad_1), where the adversary makes a query (y, P^{-1}) before Q_k, we note that, y is uniformly sampled by the system, and unless the adversary knows a secret key SK_y such that $y = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_y))$, $(y, \tilde{P}^{-1}(y))$ are independent of $\mathcal{A}$'s view. Applying Lemma 12, we have

$$\forall \mathsf{SK} \in \mathcal{X}, \Pr[\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})) = y] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}},$$

which refers to

$$\Pr[\mathcal{A} \text{ knows } \mathsf{SK}_y] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

And under the condition that $\mathcal{A}$ does not know SK_y, the probability that the adversary makes a query (y, P^{-1}) before Q_k is trivially bounded $\frac{k-1}{|\mathcal{Y}|}$. Thus, we have

$$\Pr[\mathsf{Bad}_1] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{k-1}{|\mathcal{Y}|}.$$

For the bad event on H_1 query(Bad_2), where during the game the adversary makes a query $(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1); H_1)$, we note that, to do so, the adversary has to output a valid shared key $\mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1))$. Moreover, if $\tilde{P}(\mathsf{PK}^*)$ is known by the adversary, then this bad event occurs with high probability. Fortunately, we can argue that $\tilde{P}(\mathsf{PK}^*)$ is independent of $\mathcal{A}$ except for a negligible probability. Essentially, $\mathcal{A}$ cannot learn $\tilde{P}(\mathsf{PK}^*)$ by just making P query(by definition $P^{\mathsf{r}}(\mathsf{PK}^*) = y$, rather than $\tilde{P}(\mathsf{PK}^*)$), and an alternative way to learn $\tilde{P}(\mathsf{PK}^*)$ is that, the adversary knows a secret key SK^* such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$, and it outputs $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*))$. However, due to definition, we know that $\mathsf{SCC}_k = 1$, which means the adversary never makes $(\mathsf{SK}^*; H_0)$ or $(\mathsf{SK}^*; \mathsf{NKG})$ in the previous queries. Besides, $\tilde{H}_0$ is a random oracle, it's apparent that for any secret key SK, $\tilde{H}_0(\mathsf{SK})$ is uniformly distributed in $\mathcal{X}$. Applying Lemma 12, we have

$$\forall \mathsf{SK} \in \mathcal{X}, \Pr[\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})) = \tilde{P}(\mathsf{PK}^*)] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}},$$

which refers to

$$\Pr[\mathcal{A} \text{ knows } \mathsf{SK}^*] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Hence, $\tilde{P}(\mathsf{PK}^*)$ is independent of the adversary's view except for negligible probability. Moreover, due to the entropic shared keys, we have that, under the condition that $\tilde{P}(\mathsf{PK}^*)$ is independent of $\mathcal{A}$'view, the probability that $\mathcal{A}$ outputs a valid shared key is bounded by:

$$\Pr[\mathcal{A} \text{ outputs a valid shared key} | \tilde{P}(\mathsf{PK}^*) \text{ is independent of } \mathcal{A}\text{'s view}] \leq q\epsilon_2.$$

Thus, we have

$$\Pr[\mathsf{Bad}_2] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + q\epsilon_2.$$

Next, we prove that, with high probability, the consistency conditions also hold.

First equation. Under the condition that the responses of P and P^{-1} are consistent, it's trivial that the first equation holds.

Second equation. For one direction, where the adversary attempts to outputs $\mathsf{pk}_1 \neq \mathsf{pk}_2$ such that $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$, we note that after Q_k, the system sets $P^{-1\mathsf{r}}(y) = \mathsf{PK}^*$ and the system always responds with PK^* to the query $(\tilde{P}(\mathsf{PK}^*), P^{-1})$. Hence, the only case that $\mathcal{A}$ would break this equation is $\mathcal{A}$ makes a query $(\tilde{P}(\mathsf{PK}^*); P^{-1})$. Hence, this bad event can be trivially bounded by Bad_2.

For the other direction, where the adversary attempts to outputs $\mathsf{PK}_1 \neq \mathsf{PK}_2$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_2)$, we observe that, the response of a P query $Q_i = (\mathsf{PK}; P)$ is either $\tilde{P}(\mathsf{PK})$ (case 1 and 3) or a random string in $\mathcal{Y}$ (case 2), hence there are only two cases that $\mathcal{A}$ would break this equation: 1) there is a collision of the random strings sampled by the system(case 2); 2) the adversary makes a query $Q_j = (\tilde{P}^{-1}(y); P)$ where $Q_j \in H_0 \cup P \cup P^{-1}$ or $\mathsf{SCC}_j = 0$(case 1, 3). The probability of the first case is, of course, bounded by $\frac{q^2}{|\mathcal{Y}|}$, and for the second one, the probability is negligible if $(\tilde{P}^{-1}(y), y)$ are independent of $\mathcal{A}$'s view, which is bounded by Bad_1.

Third equation. It's trivial that the only case that adversary can break the equation is that $\mathcal{A}$ knows a secret key SK_y such that $\mathsf{keygen}(\tilde{H}_0)(\mathsf{SK}_y) = y$. Specifically, on the left side,

$$\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_y) = \widetilde{\mathsf{NKG}}(\mathsf{SK}_y) = \tilde{P}^{-1}(y),$$

while on the right side,

$$P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_y))) = P^{-1\mathsf{r}}(y) = \mathsf{PK}^*.$$

By the analysis for Bad_1, trivial to note that this equation hold unless Bad_1 occurs.

Fourth equation. By definition, it's apparent that after Q_k and $Q_i = (\mathsf{SK}_1, H_0)$, the system implicitly sets

$$H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}^*), H_0^{\mathsf{r}}(\mathsf{SK}_1)) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1),$$

and before Q_i or Q_k, the system responds this query with

$$\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(y, \tilde{H}_0(\mathsf{SK}_1)) \neq \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1).$$

Hence it's rest to prove that before Q_k or Q_i, the adversary cannot output a valid shared key $\mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}^*), H_0^{\mathsf{r}}(\mathsf{SK}_1))$ except for negligible probability.

BEFORE Q_k. In this case $\mathcal{A}$ has to output a valid shared key before Q_k. And unless $\mathsf{Bad_1}$ occurs, y is independent of $\mathcal{A}$'s view, which means the adversary is obligated to output a valid shared key without knowing one party's public key y. Moreover, due to pseudorandom public keys, y is computational close to a random public key, and then applying the entropic shared keys, we can bound the bad event here by $q(\Pr[\mathsf{Bad_1}] + \epsilon_2 + \epsilon_3)$.

BEFORE Q_i. In this case $\mathcal{A}$ has to output a valid shared key before Q_i, and we immediately observe that $\mathcal{A}$ might knows y and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK_1}))$($\mathcal{A}$ makes queries $(\mathsf{SK_1}; \mathsf{NKG})$ and $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK_1}); P)$). Moreover, $\mathcal{A}$ has access to all interfaces and those information might help it predict the shared key, and we denote $\mathsf{Bad_3}$ as the event that $\mathcal{A}$ outputs a valid shared key for $(y, \mathsf{keygen}(\tilde{H}_0(\mathsf{SK_1})))$. First, note that y is uniformly distributed and due to pseudorandom public key, $\Pr[\mathsf{Bad_3}] \leq \Pr[\overline{\mathsf{Bad_3}}] + \epsilon$, where $\overline{\mathsf{Bad_3}}$ denotes the event that that $\mathcal{A}$ outputs a valid shared key for $(\mathsf{keygen}(\bar{\mathsf{sk}}), \mathsf{keygen}(\tilde{H}_0(\mathsf{SK_1})))$ where $\bar{\mathsf{sk}} \leftarrow \mathcal{X}$. Next, we prove that $\Pr[\overline{\mathsf{Bad_3}}] \leq q^2\epsilon_1$. Essentially, assuming $\mathcal{A}$ is an adversary breaks $\overline{\mathsf{Bad_3}}$ with advantage ϵ, then we can build a reduction R that breaks semi-active unpredictable shared keys with advantage $\frac{\epsilon}{q^2}$, even the adversary knows a secret key SK^* such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$ and it does not makes a query $(\mathsf{SK}^*; H_0)$ before it outputs the shared key, which only strengthen our result. Concretely, let $\mathcal{C}$ be the challenger of semi-active unpredictable shared keys with challenge public key $(\mathsf{pk_1}, \mathsf{pk_2})$ and the corresponding oracles $(\mathcal{O}_1, \mathcal{O}_2)$, then the reduction $R^{\mathcal{A}}$ works as follows:

- R simulates(lazy sampler) four random oracles $H'_0, P', P^{-1'}$ and H'_1 itself.
- R simulates NKG and NSK just by the oracles. For instance, $\mathsf{NKG}(\mathsf{SK}) = P^{-1'}(\mathsf{keygen}(H'_0(\mathsf{SK})))$.
- R simulates H_0 queries by the H'_0 and if the adversary makes queries $(\mathsf{SK}^*; H_0)$ or $(\mathsf{SK_1}; H_0)$, our reduction aborts;
- R simulates P queries and P^{-1} queries as the system in Game 2 by using P' and $P^{-1'}$, except that it implicitly sets $P(\mathsf{PK}^*) = \mathsf{pk_1}$ and $P(\mathsf{NKG}(\mathsf{SK_1})) = pk_2$.
- For H_1 queries, R use both H'_1 and the oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Specifically, let $(\mathsf{PK}, \mathsf{PK}', \mathsf{shk}; H_1)$ be the query:
  1. If $\mathsf{PK} \notin \{\mathsf{PK}^*, \mathsf{PK_1}\}$ and $\mathsf{PK}' \notin \{\mathsf{PK}^*, \mathsf{PK_1}\}$, then R responds to it as the system in Game 2 by only using H_1.
     Note that the reduction controls every interfaces, and once the adversary makes a $\mathsf{NKG}(\mathsf{SK})$ query, $H_0(\mathsf{SK})$ is known by R.
  2. If $\mathsf{PK} \in \{\mathsf{PK}^*, \mathsf{PK_1}\}$ and $\mathsf{PK}' \in \{\mathsf{PK}^*, \mathsf{PK_1}\}$, then R responds to it as the system in Game 2 by only using H_1.
  3. If $\mathsf{PK} = \mathsf{PK}^* \wedge \mathsf{PK}' \notin \{\mathsf{PK}^*, \mathsf{PK_1}\}$, and $P(\mathsf{PK}')$ is known by R, then it makes a call $(P(\mathsf{PK}'), \mathsf{shk})$ to $\mathcal{O}_1$. If $\mathcal{O}_1$ answers 1, then R responds to the query with $\mathsf{NSK}(\mathsf{PK}', \mathsf{SK}^*)$, otherwise as the system in Game 2 by only using H_1.
  4. If $\mathsf{PK}' = \mathsf{PK}^* \wedge \mathsf{PK} \notin \{\mathsf{PK}^*, \mathsf{PK_1}\}$, and $P(\mathsf{PK})$ is known by R, then it makes a call $(P(\mathsf{PK}), \mathsf{shk})$ to $\mathcal{O}_1$. If $\mathcal{O}_1$ answers 1, then R responds to the query with $\mathsf{NSK}(\mathsf{PK}, \mathsf{SK}^*)$, otherwise as the system in Game 2 by only using H_1.

5. If $\mathsf{PK} = \mathsf{PK}_1 \wedge \mathsf{PK}' \notin \{\mathsf{PK}^*, \mathsf{PK}_1\}$, and $P(\mathsf{PK}')$ is known by R, then it makes a call $(P(\mathsf{PK}'), \mathsf{shk})$ to $\mathcal{O}_2$. If $\mathcal{O}_2$ answers 1, then R responds to the query with $\mathsf{NSK}(\mathsf{PK}', \mathsf{SK}_1)$, otherwise as the system in Game 2 by only using H_1.
6. If $\mathsf{PK}' = \mathsf{PK}_1 \wedge \mathsf{PK} \notin \{\mathsf{PK}^*, \mathsf{PK}_1\}$, and $P(\mathsf{PK})$ is known by R, then it makes a call $(P(\mathsf{PK}), \mathsf{shk})$ to $\mathcal{O}_2$. If $\mathcal{O}_2$ answers 1, then R responds to the query with $\mathsf{NSK}(\mathsf{PK}, \mathsf{SK}_1)$, otherwise as the system in Game 2 by only using H_1.

Trivially note that, the view of $\mathcal{A}$ in Game 2 is identical to the view that simulated by the reduction[14] and R can just outputs one of the shared key shk when the adversary makes a query $(\mathsf{PK}^*, \mathsf{PK}_1, \mathsf{shk}; H_1)$. As the reduction randomly sets the challenging public key(at most q choices), and shk is the one of the shared key(at most q queries) that $\mathcal{A}$ outputs, hence $\Pr[R^{\mathcal{A}} \text{ wins}] \geq \frac{\Pr[\overline{\mathsf{Bad}_3}]}{q^2}$, referring to $\Pr[\mathsf{Bad}_3] \leq \epsilon_3 + q^2\epsilon_1$. Combing together, we have

$$\Pr[\text{Bad events for 4th equation}] \leq q(\Pr[\mathsf{Bad}_1 + \epsilon_2 + \epsilon_3 + q^2\epsilon_1 + \epsilon_3]).$$

Fifth equation. Easy to note that the 1 st and 3rd equation imply the fifth one.

Sixth equation. By definition, we immediately observe that the query $Q_j = (\mathsf{NKG}^r(\mathsf{SK}_2); P)$ satisfies $\mathsf{SCC}_j = 0$, otherwise, the adverary knows nothing of SK_2 except for negligible probability ($\leq q \Pr[\mathsf{Bad}_1]$). Besides, if the adversary makes a query $(\mathsf{SK}_2; H_0)$ before this H_1 query, the 6th equation holds for free as the system would follow case 3.2 to responds to this H_1 query. Hence the only case that the adversary would break the equation is that, under the condition that $\mathsf{SCC}_j = 0$ and no $((\mathsf{SK}_2; H_0))$ before, the adversary outputs a valid shared key $\mathsf{shk} = \mathsf{sharedkey}(P^r(\mathsf{NKG}^r(\mathsf{SK}_2)), \mathsf{sk}_1)$ where $\mathsf{keygen}(\mathsf{sk}_1) = y$. Applying the same analysis above (case "before Q_i" in the fourth equation), we can bound this bad event by $\Pr[\mathsf{Bad}_3]$.

Seventh equation. Easy to observe that, the bad events that break this question are identical to the ones in the fourth equation, hence this equation holds as long as the 4th equation holds.

Eighth equation. We immediately note that, in this equation, every query is either a H_0 query or a NKG query or a NSK query. Moreover, the response of those queries are identical in both games, hence this equation holds for free.

Combing together, we can bound the unoin of the bad events as

$$\Pr[\mathsf{Bad}] \leq \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_3] + \epsilon_2 + 2\epsilon_3 + q^2\epsilon_1,$$

which refers to

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

[14] the purpose of $\mathcal{O}_1$ and $\mathcal{O}_2$ is to makes sure that, for H_1 query, the case 2.2 and case 2.3 can be answered correctly. And once the system has the power to check the validity of the shared key, say after Game 5, the reduction would not use the oracle anymore.

Claim. Game 2 $\approx$ Game 3.

Proof. Recalling the only difference between Game 3 and Game 2 is that the system replace the responses of the P query (case 2) with a random public key rather than a random string. Specifically, let $Q_k = (\mathsf{PK}^*, H_0)(Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1)$, in Game 3, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds with $\mathsf{pk} = \mathsf{keygen}(\mathsf{sk})$, and inserts $(*, *, \mathsf{pk}, \mathsf{PK}^*)$ into the P-table.

The proof here is straightforward, as the system just change the distribution of the responses from random strings to random public keys and the system would not insert the secert key sk into P-table, so there is a trivial reduction from distinguisher of Game 2 and Game 3 to the pseudorandom public key property[15]. Therefore, we have that

$$|\Pr[\text{Game 2}] - \Pr[\text{Game 3}]| \leq q^2\epsilon_3 \leq \mathsf{negl}(\lambda).$$

Claim. Game 3 $\approx$ Game 4.

Proof. Recalling that the only difference between Game 4 and Game 3 is that the system also records the secret key sk into P-table. More concretely, let $Q_k = (\mathsf{PK}^*, H_0)(Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1)$, in Game 4, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with $\mathsf{pk} = \mathsf{keygen}(\mathsf{sk})$, and inserts $(*, \mathsf{sk}, \mathsf{pk}, \mathsf{PK}^*)$ into the P-table. Hence, the only case to distinguish Game 4 from Game 3 is when sk is used. In fact, unless the adversary makes a query $(\mathsf{SK}^*; H_0)$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$, the responses of all queries in both games are identical. Applying the analysis above (Game 1$\approx$ Game 2), it's trivial that

$$\Pr[\mathcal{A} \text{ outputs } \mathsf{SK}^*] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{Y}|}$$

which refers to

$$|\Pr[\text{Game 3}] = \Pr[\text{Game 4}]| \leq q(q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{Y}|}) \leq \mathsf{negl}(\lambda).$$

Claim. Game 4 $\approx$ Game 5.

Proof. Recalling that the only difference between Game 4 and Game 5 occurs in case 3, where $Q_k = (\mathsf{PK}^*; P) \notin H_0 \cup P \cup P^{-1}$ but $\mathsf{SCC}_k = 0$. Therefore the adversary knows at least one secret key SK^* such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{PK}^*$. The fact is that in Game 4, the system responds to Q_k with $\tilde{P}(\mathsf{PK}^*)$ while in Game 5 the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with $\mathsf{keygen}(\mathsf{sk})$ and inserts $(*, \mathsf{sk}, \mathsf{keygen}(\mathsf{sk}), \mathsf{PK}^*)$ into P table.

[15] It's important that the system does not use sk(say, inserts it into P-table) as in the reduction, the simulated system has no access to sk. And that's the reason why we need an additional game (Game 4) to complete the proof.

Analogous to the analysis above(Game 1 $\approx$ Game 2), we know that, under the condition that the bad events above never occur, $\mathcal{A}$'s view on Game 4 is: the responses of H_0 and H_1 are independent and random strings. For P queries $Q_k = (\mathsf{PK}; P) \notin H_0 \cup P \cup P^{-1}$, the response is a random public key, no matter with $\mathsf{SCC}_k = 0$ or 1. For P^{-1} query $Q = (\mathsf{pk}; P^{-1})$, the response should be the inverse of P, and if $Q \notin H_0 \cup P \cup P^{-1}$, then the response is independent and uniformly distributed. Moreover, the responses also satisfy the following equations:

1. $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk})) = \mathsf{pk}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{PK})) = \mathsf{PK}$;
2. There exists no $(\mathsf{PK}_1 \neq \mathsf{PK}_2), (\mathsf{pk}_1 \neq \mathsf{pk}_2)$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_2)$ or $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$;
3. $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK})))$;
4. $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1^{r}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$;
5. $P^{r}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}))$(case 2 for H_0 query);
6. $H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)), \mathsf{sk}_1))$
   $= \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2)$ (case 2.2, 2.3 for H_1 query) ;
7. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{PK}_2, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_2), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$ (case 3.2, 3.3 for H_1 query);
8. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$
   $= \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{SK}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{SK}_1)$(case 3.1 for H_1 query) .

Now we turn to Game 5, where the system responds to $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$ with a random public key(the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$ and responds with $\mathsf{pk} = \mathsf{keygen}(\mathsf{sk})$), comparing to $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ in Game 4. For H_0 query, after Q_k, the system implicitly sets $H_0^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{sk}$, as sk is sampled uniformly, the response is well-distributed. For P query, as pk is a random public key, the response is also well-distributed. After Q_k, the system implicitly sets $P^{-1\mathsf{r}}(\mathsf{pk}) = \mathsf{PK}^*$, thus as long as the adversary never sees pk before(say $\mathcal{A}$ makes a query $(\mathsf{pk}; P^{-1})$), the responses of P^{-1} queries are well-formed and consistent. In fact, pk is a random public key, due to pseudorandom public key property, we can bound this bad event by $\frac{k-1}{|\mathcal{Y}|} + (k-1)\epsilon_3$. For H_1 query, by definition, we have that after Q_k, the system implicitly sets:

$$\begin{aligned}
&H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\mathsf{pk}, H_0^{\mathsf{r}}(\mathsf{SK}_1)) \\
=&H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\mathsf{sk}, P^{r}(\mathsf{PK}_1)) \\
=&\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1) = \tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1));
\end{aligned}$$

where $\mathsf{PK}_1 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1)$. Hence, as long as the adversary does not make a query $(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1); H_1)$, the responses of H_1 queries are well distributed (as $\tilde{H}_1$ of itself is a random oracle). Same as above, we note that, if such a bad event occurs, the adversary is obligated to output a valid shared key $\mathsf{sharedkey}(\tilde{P}(\mathsf{PK}^*), \tilde{H}_0(\mathsf{SK}_1))$, where $\mathcal{A}$ knows $\tilde{H}_0(\mathsf{SK}_1)$($\mathcal{A}$ can just make a query $(\mathsf{SK}_1; H_0)$). Next, we show that, even $\mathcal{A}$ knows SK^*, $\tilde{P}(\mathsf{PK}^*)$ is still independent of $\mathcal{A}$'s view. Firstly, it's trivial that the adversary cannot knows $\tilde{P}(\mathsf{PK}^*)$ by

just making P query, secondly $Q_k \notin H_0$ which means the adversary never makes query $(\mathsf{SK}^*; H_0)$ before Q_k and after Q_k, $H_0^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{sk}$. Hence $\tilde{H}_0(\mathsf{SK}^*)$ is hidden from adversary's view(the best strategy to guess $\tilde{P}(\mathsf{PK}^*)$ is randomly samples $x \xleftarrow{\$} \mathcal{X}$ and outputs $\mathsf{keygen}(x)$), which implies

$$\Pr[\mathcal{A} \text{ outputs } \tilde{P}(\mathsf{PK}^*)] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Next, under the condition that $\tilde{P}(\mathsf{PK}^*)$ is independent, the probability that $\mathcal{A}$ outputs a valid shared key is bound by the entropic shared key property and pseudoranom public key property, specifically,

$$\Pr[\mathcal{A} \text{ outputs a valid shared key} | \tilde{P}(\mathsf{PK}^*) \text{ is independent}] \leq q(\epsilon_2 + \epsilon_3).$$

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

First equation. Under the condition that the responses of P and P^{-1} are consistent, it's trivial that the first equation holds.

Second equation. For one direction, where the adversary attempts to outputs $\mathsf{pk}_1 \neq \mathsf{pk}_2$ such that $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$, we note that after Q_k, the system sets $P^{-1\mathsf{r}}(\mathsf{pk}) = \mathsf{PK}^*$ and the system always responds with PK^* to the query $(\tilde{P}(\mathsf{PK}^*), P^{-1})$. Hence, the only case that $\mathcal{A}$ would break this equation is $\mathcal{A}$ makes a query $(\tilde{P}(\mathsf{PK}^*); P^{-1})$. Fortunately, we've already proven that

$$\Pr[\mathcal{A} \text{ outputs } \tilde{P}(\mathsf{PK}^*)] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

For the other direction, where the adversary attempts to outputs $\mathsf{PK}_1 \neq \mathsf{PK}_2$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_1)$, we observe that, the response a P query $Q_i = (\mathsf{PK}; P)$ is either $\tilde{P}(\mathsf{PK})$ (case 1) or a random public key (case 2 and 3), hence there are only two cases that $\mathcal{A}$ would break this equation: 1) there is a collision of the random public key sampled by the system; 2) the adversary makes a query $Q_j = (\tilde{P}^{-1}(\mathsf{pk}); P)$ where $Q_j \in H_0 \cup P \cup P^{-1}$ or $\mathsf{SCC}_j = 0$. The probability of the first case is bounded by(applying Lemma 12 and pseudorandom public key) $q^2\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2}{|\mathcal{Y}|} + q\epsilon_3$, and for the second one, we note that pk is a random public key, and $\mathcal{A}$ learns nothing of $\tilde{P}^{-1}(y)$ by making a query $(y; P^{-1})$, which means $\tilde{P}^{-1}(y)$ is hidden from $\mathcal{A}$. Combining together, we can bound the union of those bad events by:

$$\Pr[\text{Bad events for 2nd equation}] \leq q^2\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2}{|\mathcal{Y}|} + q\epsilon_3 + \frac{q}{|\mathcal{Y}|}.$$

Third equation. By definition, we have that after Q_k, the system implicitly sets $H_0^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{sk}; P^{-1\mathsf{r}}(\mathsf{pk}) = \mathsf{PK}^*$. The potential attack to break this equation

is that, $\mathcal{A}$ outputs $\mathsf{SK_{pk}}$ such that $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK_{pk}})) = \mathsf{pk}$. Specifically, on the left side

$$\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK_{pk}}) = \widetilde{\mathsf{NKG}}(\mathsf{SK_{pk}}) = \tilde{P}^{-1}(\mathsf{pk}),$$

while on the right side,

$$P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK_{pk}}))) = P^{-1\mathsf{r}}(y) = \mathsf{PK}^*.$$

Now we bound this bad event as follows: as pk is a random public key, and due to pseudorandom public key property, we have that in the previous H_0 query or NKG query(say $Q_i = (\mathsf{SK_i}; H_0)$),$\Pr[\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK_i})) = \mathsf{pk}] = \frac{1}{|\mathcal{Y}|} + \epsilon_3$. Hence the probability that no "good" $\mathsf{SK_i}(i < k)$ is bounded by $\frac{k-1}{|\mathcal{Y}|} + (k-1)\epsilon_3$. Under the condition that no good $\mathsf{SK_i}$ appears before Q_k, applying $\tilde{H}_0$ is a random oracle again, we have that after Q_k, the adversary cannot output $\mathsf{SK_{pk}}$ except for probability (applying Lemma 12)

$$\Pr[\mathcal{A} \text{ outputs } \mathsf{SK_{pk}}] \leq (q-k)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Thus we can bound the bad event that breaks 3rd equation by

$$\Pr[\text{Bad events for 3rd equation}] \leq \frac{k-1}{|\mathcal{Y}|} + (k-1)\epsilon_3 + (q-k)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Fourth equation. By definition, it's apparent that after Q_k and $Q_i = (\mathsf{SK}_1, H_0)$, the system implicitly sets

$$H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}^*), H_0^{\mathsf{r}}(\mathsf{SK}_1)) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1),$$

and before Q_i or Q_k, the system responds this query with

$$\tilde{H}_1(\mathsf{PK}_1, \mathsf{PK}^*, \mathsf{sharedkey}(\mathsf{pk}, H_0^{\mathsf{r}}(\mathsf{SK}_1)) \neq \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{SK}_1).$$

Hence it's rest to prove that before Q_k or Q_i[16], the adversary cannot output a valid shared key $\mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}^*), H_0^{\mathsf{r}}(\mathsf{SK}_1))$ except for negligible probability.

Before Q_k. In the case that $\mathcal{A}$ outputs a valid shared key before Q_k, we note that the adversary is obligated to output a valid shared key without knowing one party's public key pk. As pk is a random public key, it's trivial that this case is bounded by the entropic shared key property and pseudorandom public key property.

Before Q_i. In this case that $\mathcal{A}$ outputs a valid shared key before Q_i, we note that $\mathcal{A}$ might knows pk and $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1))$($\mathcal{A}$ makes queries $(\mathsf{SK}_1; \mathsf{NKG})$ and $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1); P)$), and $\mathcal{A}$ is able to output a valid shared key by only having two

[16] due to $Q_k \notin H_0$, it's trivial that the query $(\mathsf{SK}^*; H_0)$ is always after Q_k, which means the adversary can not getch $P^{\mathsf{r}}(\mathsf{PK}^*)$ by calculatting $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*))$ before Q_k.

public keys $(\mathsf{pk}, \mathsf{keygen}(H_0^r(\mathsf{SK}_1)))$). Hence, applying exactly the same technique as above(the case "before Q_i" in Game 1 $\approx$ Game 2), we can bound this bad case by the semi-active unpredictable shared key property. Combing together, we have

$$\Pr[\text{Bad events for 4th equation}] \leq q(q^2\epsilon_1 + \epsilon_2 + \epsilon_3).$$

Fifth equation. Easy to note that the 1 st and 3rd equation imply the fifth one.

Sixth equation. By definition, we immediately observe that the query $Q_j = (\mathsf{NKG}^r(\mathsf{SK}_2); P)$ satisfies $\mathsf{SCC}_j = 0$, otherwise, the adverary knows nothing of SK_2 except for negligible probability ($\leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{Y}|}$). Besides, if the adversary makes a query $(\mathsf{SK}_2; H_0)$ before this H_1 query, the 6th equation holds for free as the system would follow case 3.2 to responds to this H_1 query. Hence the only case that the adversary would break the equation is that, under the condition that $\mathsf{SCC}_j = 0$ and no $((\mathsf{SK}_2; H_0))$ before, the adversary outputs a valid shared key $\mathsf{shk} = \mathsf{sharedkey}(P^r(\mathsf{NKG}^r(\mathsf{SK}_2)), \mathsf{sk}_1)$ where $\mathsf{keygen}(\mathsf{sk}_1) = \mathsf{pk}$.

As the adversary makes this H_1 query before $(\mathsf{SK}_2; H_0)$, the response of $(\mathsf{NKG}^r(\mathsf{SK}_2); P)$ is another public key, and the adversary knows nothing of the corresponding secret key. Hence, the probability that $\mathcal{A}$ outputs a valid shared key is bounded by the semi-active unpredictable shared keys. Combing together, we have:

$$\Pr[\text{Bad events for 6th equation}] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{Y}|} + q^3\epsilon_1.$$

Seventh equation. Easy to observe that, the bad events that break this equatoin are identical to the ones in the fourth equation, hence this equation holds as long as the 4th equation holds.

Eighth equation. By definition, we note that in Game 5, $\mathsf{keygen}(H_0^r(\mathsf{SK}_1)) = P^r(\mathsf{NKG}^r(\mathsf{SK}_1))$, which means that this equation holds if the 7th equation holds.

Combing together, we can bound the unoin of the bad events as

$$\Pr[\mathsf{Bad}] \leq (q^2{+}2q{-}k)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}{+}2q(q^2\epsilon_1{+}\epsilon_2{+}\epsilon_3){+}(q{+}2k{-}2)\epsilon_3{+}\frac{q^2+2q+2(k-1)}{|\mathcal{Y}|},$$

which refers to

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

Claim. Game 5 $\approx$ Game 6.

Proof. Recalling that the only difference betwwen Game 5 and Game 6 occurs in case 2(answering P^{-1} query), where $Q_k = (\mathsf{pk}^*; P^{-1}) \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$. Therefore, there is no $(\mathsf{SK}^*; \mathsf{NKG})$ such that $\mathsf{keygen}(H_0^r(\mathsf{SK}^*)) = \mathsf{pk}^*$ before Q_k. In Game 5, the system responds to Q_k with $\tilde{P}^{-1}$ while in Game 6, it

samples $\mathsf{SK} \xleftarrow{\$} \mathcal{X}$, responds with $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$, and inserts $(\mathsf{SK}, *, \mathsf{pk}^*, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ into P^{-1} table.

Similar to the analysis above, we note that, under the condition that the bad events above never occur, $\mathcal{A}$'s view on Game 5 is: the responses of H_0 and H_1 are independent and random strings. For P queries $Q_k = (\mathsf{PK}; P) \notin H_0 \cup P \cup P^{-1}$, the response is a random public key. For P^{-1} query, the response should be the inverse of P, and if $Q \notin H_0 \cup P \cup P^{-1}$, then the response is independent and uniformly distributed. Moreover, the responses also satisfy the following equations:

1. $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk})) = \mathsf{pk}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{PK})) = \mathsf{PK}$;
2. There exists no $(\mathsf{PK}_1 \neq \mathsf{PK}_2), (\mathsf{pk}_1 \neq \mathsf{pk}_2)$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_2)$ or $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$;
3. $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK})))$;
4. $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1^{r}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$;
5. $P^{r}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}))$(case 2 for H_0 query);
6. $H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)), \mathsf{sk}_1))$
   $= \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2)$ (case 2.2, 2.3 for H_1 query) ;
7. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{PK}_2, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_2), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$ (case 3.2, 3.3 for H_1 query);
8. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$
   $= \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{SK}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{SK}_1)$(case 3.1 for H_1 query) .

Now we analyze the view on the adversarial interfaces. For H_0 query, the responses in both games are identical. For P query, the system implicitly sets $P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{pk}^*$ after Q_k, thus as long as the adversary never sees $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ before(say $\mathcal{A}$ makes a query $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}); P)$ before Q_k), the response of P queries are well distributed and consistent. In fact, SK is sampled uniformly from $\mathcal{X}$, and applying Lemma 12, we have

$$\forall z, \Pr_{\mathsf{SK} \xleftarrow{\$} \mathcal{X}}[\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = z] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}},$$

which means

$$\Pr[\mathcal{A} \text{ outputs } \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})] \leq (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

For P^{-1} query, we know that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = \tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})))$. Besides, applying Lemma 12, it's apparent that,

$$\forall \mathsf{pk}, \Pr_{\mathsf{SK} \xleftarrow{\$} \mathcal{X}}[\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})) = \mathsf{pk}] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}},$$

hence, in $\mathcal{A}$'s view, $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ is an independent and random string except for negligible probability($\leq (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}$). For H_1 query, we note that after

Q_k, the system implicitly sets:

$$\begin{aligned}
&H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{shk}_1)\\
=&\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}) \text{ (case 2.2, 2.3)}\\
=&\tilde{H}_1(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{sharedkey}(\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})), \tilde{H}_0(\mathsf{SK}_1));\\
&H_1^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_2)))\\
=&\mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{SK}_2) \text{ (case 3.2, 3.3)}\\
=&\tilde{H}_1(\mathsf{PK}_2, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{sharedkey}(\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})), \tilde{H}_0(\mathsf{SK}_2)).
\end{aligned}$$

where shk_1[17]is the valid shared key between pk^* and $P^{\mathsf{r}}(\mathsf{PK}_1)$ and $\mathsf{PK}_\ell = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_\ell)$. Therefore, unless $\mathcal{A}$ calls $(\mathsf{PK}_\ell, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{sharedkey}(\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})), \tilde{H}_0(\mathsf{SK}_\ell); H_1)$, the responses of H_1 queries are well distributed (as $\tilde{H}_1$ of itself is a random oracle). Same as above, we note that, if such a bad event occurs, the adversary is obligated to output a valid shared key $\mathsf{sharedkey}(\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})), \tilde{H}_0(\mathsf{SK}_\ell)$, where $\mathcal{A}$ might know $\tilde{H}_0(\mathsf{SK}_1)$($\mathcal{A}$ can just make a query $(\mathsf{SK}_\ell; H_0)$). However, $\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ is independent of $\mathcal{A}$'s view, in fact, easy to note that, except for randomly guessing(the best strategy is that $\mathcal{A}$ samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$ and outputs $\mathsf{keygen}(\mathsf{sk})$, so the probability is bounded by $q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}$), the only case $\mathcal{A}$ can output $\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ is $\mathcal{A}$ makes a query $(\mathsf{SK}; H_0)$(for P query Q, in either game, the system only responds with $\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ in case 1, where $Q \in H_0$). However, SK is sampled uniformly by the system, hence, we have

$$\Pr[\mathcal{A} \text{ outputs } \tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{X}|}.$$

Now, under the condition that $\tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}))$ is independent, the probability that $\mathcal{A}$ outputs a valid shared key is trivially bounded by the entropic shared key property, specifically,

$$\Pr[\mathcal{A} \text{ outputs a valid shared key} | \tilde{P}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) \text{ is independent}] \leq q\epsilon_2.$$

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

First equation. Under the condition that the responses of P and P^{-1} are consistent, it's trivial that the first equation holds.

Second equation. For one direction, where the adversary attempts to outputs $\mathsf{pk}_1 \neq \mathsf{pk}_2$ such that $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$, we observe that, the response of a P^{-1} query $Q_i = (\mathsf{pk}; P^{-1})$ is either $\tilde{P}^{-1}(\mathsf{pk})$ (case 1 and 3) or $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$. Hence,

[17] In fact, the adversary has two ways to calculate this shared key: 1) $\mathcal{A}$ makes a query $(\mathsf{SK}_1; H_0)$ and computes $\mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_1))$; 2) $\mathcal{A}$ makes a query $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}); P)$ and computes $\mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})), \mathsf{sk}^*)$, as pk^* is chosen by $\mathcal{A}$, A might know the secret key sk^* such that $\mathsf{keygen}(\mathsf{sk}^*) = \mathsf{pk}^*$.

under the condition that $(\mathsf{SK}; H_0)$ never appears, the only case that $\mathcal{A}$ would break this equation is: there is a collision of the response, which is bounded by:

$$\Pr[\text{Collision in } P^{-1} \text{ queries}] \leq q^2\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + q^2\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2}{|\mathcal{X}|}.$$

For the other direction, where the adversary attempts to outputs $\mathsf{PK}_1 \neq \mathsf{PK}_2$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_1)$, we observe that, the system implicitly sets $P^{\mathsf{r}}(\mathsf{NKK}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{pk}^*$, while if the adversary makes a query $(\mathsf{SK}^*; H_0)$ such that $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*)) = \mathsf{pk}^*$, then $P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*)) = \mathsf{pk}^*$. Same as above, we note that, $\tilde{H}_0$ is a random oracle, hence applying Lemma

$$\Pr_{\mathsf{SK}}[\mathsf{keygen}(\tilde{H}_0(\mathsf{SK})) = \mathsf{pk}^*] \leq \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}},$$

which refers to,

$$\Pr[\mathcal{A} \text{ outputs } \mathsf{SK}^*] \leq q\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Combing together, we can bound the union of those bad events by:

$$\Pr[\text{Bad events for 2nd equation}] \leq (2q^2 + q)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2}{|\mathcal{X}|}.$$

Third equation. By definition, we immediately note that, the responses of H_0 queries are identical in both games, hence unless the adversary makes a query $(\mathsf{SK}; H_0)$, this equation holds for free.

Fourth equation. By definition, we note that pk^* is chosen by the adversary, which means the adversary might know the corresponding secret key sk^*. Hence, the adversary can calculate the shared key $\mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_1))$ without making a query $(\mathsf{SK}_1; H_0)$, in fact, the adversary can just makes two queries $(\mathsf{SK}_1; \mathsf{NKG})$ and $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1); P))$ and outputs $\mathsf{sharedkey}(\mathsf{sk}^*, P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1)))$. However, due to both case 2.2 and 3.2 for H_1 queries, we have that, after Q_k and $Q_i = (\mathsf{SK}_1, H_0)$(case 3.2), the system implicitly sets

$$H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{PK}_1, \mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_1)) = \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{SK}_1).$$

Moreover, after Q_k and $Q_j = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1); P)$(case 2.2), the system sets

$$H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{PK}_1, \mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_1)) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}) = \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{SK}_1).$$

and before Q_k or $(Q_i \cap Q_j)$, the system responds this query with

$$\tilde{H}_1(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{PK}_1, \mathsf{sharedkey}(\mathsf{pk}^*, H_0^{\mathsf{r}}(\mathsf{SK}_1)) \neq \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{SK}_1).$$

Thus unless the adversary can make such a query before Q_k or $(Q_i \cap Q_j)$, the 4th equation holds for certain.

BEFORE Q_k. As pk^* is chosen by the adversary, sk^* is known, which means the shared key is never the barrier anymore. However, there are three components in H_1 query and one of them is $P^{-1\mathsf{r}}(\mathsf{pk}^*)$, and as we showed above, before Q_k

$$\Pr[\mathcal{A} \text{ outputs } \ \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{pk}^*)] \leq (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

BEFORE $(Q_i \cap Q_j)$. In this case, we show that, the adversary cannot outputs the valid shared key as before. In fact, the probability that $\mathcal{A}$ is able to output the corresponding shared key is immediately bounded by the entropic shared key property and , as $\mathcal{A}$ has no knowledge of the other party's public key and secret key. Combing together, we have

$$\Pr[\text{Bad events for 4th equation}] \leq q\epsilon_2 + +(k-1)\sqrt{2\epsilon_3 + \frac{1}{|X|}}.$$

Remark: Careful readers would notice that there is actually an additional case, which is $\mathcal{A}$ picks PK_1(rather than $\mathsf{PK}_1 = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1)$) and makes a query $(\mathsf{PK}_1; P)$. Meanwhile, in this case $\mathcal{A}$ knows nothing of SK_1, which means it cannot runs $\mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk}^*), \mathsf{SK}_1)$, therefore, this case would not affect the equation at all.

Fifth equation. Easy to note that the 1st and 3rd equation imply the fifth one.

Sixth equation. Easy to note that the bad events that break this equation is covered by the ones in the fourth equation.

Seventh equation. Easy to note that the bad events that break this equation is also covered by the ones in the fourth equation.

Eighth equation. Easy to note that, under the condition $\mathcal{A}$ never makes a query $(\mathsf{SK}; H_0)$, the responses of H_0 queries are identical in both games, which implies the 8th equation holds for free.

Combing together, we can bound the unoin of the bad events as

$$\Pr[\mathsf{Bad}] \leq (q^2 + 2q + 3(k-1))\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2+q}{|\mathcal{X}|} + 2q\epsilon_2,$$

which refers to

$$|\Pr[\text{Game 5}] - \Pr[\text{Game 6}]| \leq q \cdot \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Claim. Game 6 $\approx$ Game 7.

Proof. Recalling that the only diffence between Game 6 and Game 7 occurs in case 3(answering P^{-1} query), where $Q_k = (\mathsf{pk}^*; P^{-1}) \notin H_0 \cup P \cup P^{-1}$ but

$\mathsf{SCC}_k = 0$. Therefore, in such a case, $\mathcal{A}$ made a query $(\mathsf{SK}^*; \mathsf{NKG})$ before Q_k, where $\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}^*)) = \mathsf{pk}^*$.

However, by $Q_k \notin H_0$, we know that $\mathcal{A}$ does not make a query $(\mathsf{SK}^*; H_0)$ before Q_k. Moreover $Q_k \notin P$, which implies $H_0^{\mathsf{r}}(\mathsf{SK}^*) = \tilde{H}_0(\mathsf{SK}^*)$. As $\tilde{H}_0$ is a random oracle, $\tilde{H}_0(\mathsf{SK}^*)$ is unifromly distributed in $\mathcal{X}$ and hidden from $\mathcal{A}$, hence the probability that $\mathcal{A}$ can output "pk^*"(the component of Q_k) is bounded by the pseudorandom public key property, specifically,

$$|\Pr[\text{Game 6}] - \Pr[\text{Game 7}]| \leq q\epsilon_3 \leq \mathsf{negl}(\lambda).$$

Claim. Game 7 $\approx$ Game 8.

Proof. The only difference between Game 7 and Game 8 occurs in case 3(answering H_0 query), where $Q_k = (\mathsf{SK}^*; H_0) \notin H_0 \cup P$. In Game 7, the system responds with $\tilde{H}_0(\mathsf{SK}^*)$, while in Game 8, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds with sk and inserts $(\mathsf{SK}^*, \mathsf{sk}, \mathsf{keygen}(\mathsf{sk}), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*))$ into H_0 table.

Similar to the analysis above, we note that, under the condition that the bad events above never occur, $\mathcal{A}$'s view on Game 7 is: the responses of H_0 and H_1 are independent and random strings. For P queries, the response is a random public key. For P^{-1} query, the response should be the inverse of P, and if $Q \notin H_0 \cup P \cup P^{-1}$, then the response is independent and uniformly distributed. Moreover, the responses also satisfy the following equations:

1. $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{pk})) = \mathsf{pk}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{PK})) = \mathsf{PK}$;
2. There exists no $(\mathsf{PK}_1 \neq \mathsf{PK}_2), (\mathsf{pk}_1 \neq \mathsf{pk}_2)$ such that $P^{\mathsf{r}}(\mathsf{PK}_1) = P^{\mathsf{r}}(\mathsf{PK}_2)$ or $P^{-1\mathsf{r}}(\mathsf{pk}_1) = P^{-1\mathsf{r}}(\mathsf{pk}_2)$;
3. $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK})))$;
4. $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2) = H_1^r(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$;
5. $P^r(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}))$(case 2 for H_0 query);
6. $H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)), \mathsf{sk}_1))$
   $= \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2)$ (case 2.2, 2.3 for H_1 query) ;
7. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{PK}_2, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_2), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$ (case 3.2, 3.3 for H_1 query);
8. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$
   $= \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{SK}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{SK}_1)$(case 3.1 for H_1 query) .

Now we analyze the view on the adversarial interfaces. For H_0 query, as sk is uniforomly sampled, the response is well distributed. For P query, the system implicitly sets $P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*)) = \mathsf{keygen}(\mathsf{sk})$ after Q_k, moreover, easy to note that the adversary never makes a query $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*); P)$ before Q_k(otherwise $Q_k \in P$), hence the response of P queries are well distributed and consistent. For P^{-1} query, the system implicitly sets $P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk})) = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*)$, thus unless the adversary makes a query $(\mathsf{keygen}(\mathsf{sk}); P^{-1})$ before Q_k, the response is consistent. In fact, sk is sampled uniformly by the system, therefore $\mathsf{keygen}(\mathsf{sk})$ is independent of $\mathcal{A}$'s view before Q_k, which implies except for negligible probability($\leq (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}$), the adversary would not make such a

query. Moreover, in Game 8, $\tilde{H}_0(\mathsf{SK}^*)$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ are hidden from the adversary, hence $\tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ is uniformly distributed.

For H_1 query, we note that after Q_k, the system implicitly sets:

$$H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), \mathsf{sk})) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*)$$
$$= \tilde{H}_1(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), \tilde{H}_0(\mathsf{SK}^*)))$$

Therefore, unless $\mathcal{A}$ calls $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), \tilde{H}_0(\mathsf{SK}^*)); H_1)$, the responses of H_1 queries are well distributed (as $\tilde{H}_1$ of itself is a random oracle). In fact, $\tilde{H}_0(\mathsf{SK}^*)$ and $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}^*))$ are independent of $\mathcal{A}$'s view, thus the probability that $\mathcal{A}$ output a valid shared key is bounded by $q\epsilon_2$. Next, we prove that, with high probability, the consistency conditions hold.

First equation. Under the condition that the responses of P and P^{-1} are consistent, it's trivial that the first equation holds.

Second equation. Easy to note that, in Game 8, the responses of P and P^{-1} queries are independent of the real oracles(only using the tables and accessing to the honest interfaces). Specifically, the responses of P-queries are random public keys which are sampled by the system, and the responses of P^{-1} queries are $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$ where SK is either sampled by the system(case 2) or chosen by the adversary (case 1). Therefore, the second equation holds for certain unless there exist collisions in those responses, applying Lemma 12, we have that

$$\Pr[\text{Collision in} P \text{ and } P^{-1} \text{ queries}] \leq 2q^2\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Third equation. Under the condition that the responses of $P^{-1\mathsf{r}}$ are consistent, this equation holds trivially.

Fourth equation. By definition, we note that, after Q_k and $Q_i = (\mathsf{PK}_1; P)$(case 3.2), the system implicitly sets

$$H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), \mathsf{sk})) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*).$$

Moreover, after Q_k and $Q_j = (\mathsf{SK}_1, H_0)$ where $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1) = \mathsf{PK}_1$, the system sets

$$H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), \mathsf{sk})) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*).$$

However, before Q_k and $Q_i \cap Q_j$, the system responds with

$$\tilde{H}_1(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}^*), \mathsf{PK}_1, \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), \mathsf{sk})) \neq \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}^*).$$

Hence, unless the adversary can make such a query before Q_k or $(Q_i \cap Q_j)$, the 4th equation holds for certain.

BEFORE Q_k. As sk is uniformly sampled by the system, $(\mathsf{sk}, \mathsf{keygen}(\mathsf{sk}))$ are independent of $\mathcal{A}$'s view before Q_k. Hence, the probability that $\mathcal{A}$ outputs a valid shared key is bounded trivially by the entropic shared key property.

BEFORE $(Q_i \cap Q_j)$. Same as above, we have that, $(H_0^r(\mathsf{SK}_1), P^r(\mathsf{NKG}^r(\mathsf{SK}_1)))$ are independent of $\mathcal{A}$'s view. Hence, the probability that $\mathcal{A}$ outputs a valid shared key is also bounded by the entropic shared key property. Combing together, we have

$$\Pr[\text{Bad events for 4th equation}] \leq 2q\epsilon_2.$$

Fifth equation. Easy to note that the 1st and 3rd equation imply the fifth one.

Sixth equation. By definition, we note that, the response of $(\mathsf{keygen}(\mathsf{sk}_1); P^{-1})$ is identical in both game, unless the adversary makes query $(\mathsf{SK}; H_0)$ such that $\mathsf{keygen}(H_0^r(\mathsf{SK})) = \mathsf{keygen}(\mathsf{sk}_1)$, which is bounded by(applying Lemma 12) $q\sqrt{2\epsilon_3 + \frac{1}{|X|}}$.

Under the condition that the response of $(\mathsf{keygen}(\mathsf{sk}_1); P^{-1})$ is identical in both game, this equation holds as long as the 4th equation holds.

Seventh equation. Easy to note that the bad events that break this equation is also covered by the ones in the fourth equation.

Eighth equation. Easy to note that the bad events that break this equation is also covered by the ones in the fourth equation.

Combing together, we can bound the unoin of the bad events as

$$\Pr[\mathsf{Bad}] \leq (q^2 + q + (k-1))\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + 3q\epsilon_2,$$

which refers to

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

Claim. Game 8 $\approx$ Game 9.

Proof. The difference of Game 9 and Game 8 occurs in the cases associated with underline. Specifically, instead of calling the real oracle $\tilde{H}_1$, we now lazily sample the oracle using the table for H_1.

Similar to the analysis above, we note that, under the condition that the bad events above never occur, $\mathcal{A}$'s view on Game 7 is: the responses of H_0 and H_1 are independent and random strings. For P queries, the response is a random public key. For P^{-1} query, the response should be the inverse of P, and if $Q \notin H_0 \cup P \cup P^{-1}$, then the response is independent and uniformly distributed. Moreover, the responses also satisfy the following equations:

1. $P^r(P^{-1r}(\mathsf{pk})) = \mathsf{pk}, P^{-1r}(P^r(\mathsf{PK})) = \mathsf{PK}$;
2. There exists no $(\mathsf{PK}_1 \neq \mathsf{PK}_2), (\mathsf{pk}_1 \neq \mathsf{pk}_2)$ such that $P^r(\mathsf{PK}_1) = P^r(\mathsf{PK}_2)$ or $P^{-1r}(\mathsf{pk}_1) = P^{-1r}(\mathsf{pk}_2)$;
3. $\mathsf{NKG}^r(\mathsf{SK}) = P^{-1r}(\mathsf{keygen}(H_0^r(\mathsf{SK})))$;
4. $\mathsf{NSK}^r(\mathsf{PK}, \mathsf{SK}) = H_1^r(\mathsf{PK}, \mathsf{NKG}^r(\mathsf{SK}), \mathsf{sharedkey}(P^r(\mathsf{PK}), H_0^r(\mathsf{SK})))$;

5. $P^r(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})) = \mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}))$(case 2 for H_0 query);
6. $H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)), \mathsf{sk}_1))$
$= \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2)$ (case 2.2, 2.3 for H_1 query) ;
7. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{PK}_2, \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_2), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}_1)$ (case 3.2, 3.3 for H_1 query);
8. $H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(\mathsf{keygen}(H_0^{\mathsf{r}}(\mathsf{SK}_1)), H_0^{\mathsf{r}}(\mathsf{SK}_2)))$
$= \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1), \mathsf{SK}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{SK}_1)$(case 3.1 for H_1 query) .

Now we analyze the view on the adversarial interfaces. Firstly, we note that, in both games, the responses of H_0, P and P^{-1} queries are identical, which also implies the 1st, 2nd, 3rd and 5th equation hold for free. For H_1 query $Q_k = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{shk}; H_1)$, the response is well distributed unless $\mathsf{shk} = \mathsf{sharedkey}(\tilde{P}(\mathsf{PK}_1), \tilde{H}_0(\mathsf{SK}_2))$. In fact, in either game, $\tilde{H}_0, \tilde{P}$ and $\tilde{P}^{-1}$ are hidden from $\mathcal{A}$, hence due to the entropic shared key property, shk is not a valid shared key (under $\tilde{H}_0$ and $\tilde{P}$) except for negligible probability($\leq q\epsilon_2$). Next we prove that, with high probability, the consistency condition hold.

Fourth equation. By definition, we note that, after $Q_k = (\mathsf{SK}_2; H_0)$ and $Q_i = (\mathsf{PK}_1, P)$, the system implicitly sets

$$H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2),$$

besides, after $Q_k = (\mathsf{SK}_2; H_0)$ and $Q_j = (\mathsf{SK}_1; H_0)$, the system sets

$$H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}(\mathsf{PK}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2))) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{SK}_2),$$

moreover, after $Q_l = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2); P)$ and $Q_j = (\mathsf{SK}_1; H_0)$, the system sets

$$H_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2), \mathsf{sharedkey}(P^{\mathsf{r}}((\mathsf{SK}_1; H_0)), H_0^{\mathsf{r}}(\mathsf{SK}_1))) = \mathsf{NSK}^{\mathsf{r}}((\mathsf{SK}_1; \mathsf{H}_0), \mathsf{SK}_1),$$

However, before $(Q_k \cap Q_l)$ or $(Q_i \cap Q_j)$(Case A) the system responds with a random string, moreover, by definition, the system also responds with a random string if $\mathcal{A}$ makes such a query before $Q_k \cap Q_j$(Case B). Hence, unless the adversary can make such a query in either Case A or Case B.

For Case A, we note that in such a case, the adversary knows nothing of the other party's public key and secret key, hence it's trivially bounded by $q\epsilon_2$. While for Case B, the adversary might knows both public keys, however it does not know any of the two secret keys, which measn this bad case is bounded by the semi-active unpredictable shared keys. Specifically,

$$\Pr[\text{Bad events for 4th equation}] \leq q(q^2\epsilon_1 + \epsilon_2).$$

Sixth equation. By definition, we note that, after $Q_k = (\mathsf{keygen}(\mathsf{sk}_1); P^{-1})$ and $Q_i = (\mathsf{SK}_2; H_0)$, the system implicitly sets

$$\begin{aligned} &H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{PK}_2, \mathsf{sharedkey}(\mathsf{keygen}(\mathsf{sk}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2))) \\ =&\mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_2), \end{aligned}$$

besides, after Q_k and $Q_j = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2); P)$, the system sets

$$\begin{aligned} &H_1^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{PK}_2, \mathsf{sharedkey}(\mathsf{keygen}(\mathsf{sk}_1), H_0^{\mathsf{r}}(\mathsf{SK}_2)) \\ =&\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{SK}) = \mathsf{NSK}^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{keygen}(\mathsf{sk}_1)), \mathsf{SK}_1). \end{aligned}$$

where $P^{-1}(\mathsf{keygen}(\mathsf{sk}_1)) = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK})$. However, the system responds with a random string before Q_k or $(Q_i \cap Q_j)$. Therefore, as long as $\mathcal{A}$ does not make such a query before Q_k or $(Q_i \cap Q_j)$, this equation holds for certain. Applying exactly the same analysis above(Game 5 $\approx$ Game 6 - fourth equation) we have that

$$\Pr[\text{Bad events for 6th equation}] \leq q\epsilon_2 + (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}}.$$

Seventh equation. Easy to note that the bad events that break this equation is also covered by the ones in the fourth equation.

Eighth equation. Easy to note that the bad events that break this equation is also covered by the ones in the fourth equation.

Combing together, we can bound the unoin of the bad events as

$$\Pr[\mathsf{Bad}] \leq (k-1)\sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + 3q\epsilon_2 + q^3\epsilon_1,$$

which refers to

$$|\Pr[\text{Game 8}] - \Pr[\text{Game 9}]| \leq q \cdot \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Claim. Game 9 $\approx$ Game 10.

Proof. Let (F, G) be the function pair that uniformly sampled from $\mathcal{T}_{\mathsf{nike}}$(the function pair set in Definition 4, where F is a random injection and G is a random injection except $G(F(x), y) = G(F(y), x)$), and we note that the difference between Game 9 and Game 10 is that, in Game 9 the system responds to all queries by calling $(\widetilde{\mathsf{NKG}}(\cdot), \widetilde{\mathsf{NSK}}(\cdot,\cdot))$, while in Game 10 the system responds to all queries by calling $(F(\cdot), G(\cdot,\cdot))$. Next we prove that the distribution of the responses by calling $(\widetilde{\mathsf{NKG}}, \widetilde{\mathsf{NSK}})$ is close to the ones by calling (F, G).

Firstly, we note that the consistency condition holds for both pairs, more concretely, $\forall \mathsf{SK}_1, \mathsf{SK}_2, G(F(\mathsf{SK}_1), \mathsf{SK}_2) = G(F(\mathsf{SK}_2), \mathsf{SK}_1)$ and $\widetilde{\mathsf{NSK}}(\widetilde{\mathsf{NKG}}(\mathsf{SK}_1), \mathsf{SK}_2) = \widetilde{\mathsf{NSK}}(\widetilde{\mathsf{NKG}}(\mathsf{SK}_2), \mathsf{SK}_1)$.

Secondly, $\tilde{P}$ is a random permutation model and $\tilde{H}_1$ is a random oracle. For NKG queries, unless the adversary outputs $\mathsf{SK}_1 \neq \mathsf{SK}_2$ but $\mathsf{keygen}(\tilde{H}_0(\mathsf{SK}_1)) = \mathsf{keygen}(\tilde{H}_0(\mathsf{SK}_2))$, the responses are independent and injective. For NSK queries, under the condition that no collision in NKG, the responses are independent and injective (under such shared key condition) except that the adversary outputs

$(\mathsf{PK}_1, \mathsf{SK}_2)$ and $(\mathsf{PK}'_1, \mathsf{SK}'_2)$ such that

$$(\mathsf{PK}_1, \widetilde{\mathsf{NKG}}(\mathsf{SK}_2), \mathsf{shk}_1) \neq (\mathsf{PK}'_1, \widetilde{\mathsf{NKG}}(\mathsf{SK}'_2), \mathsf{shk}_2)$$
$$\tilde{H}_1(\mathsf{PK}_1, \widetilde{\mathsf{NKG}}(\mathsf{SK}_2), \mathsf{shk}_1) = \tilde{H}_1(\mathsf{PK}'_1, \widetilde{\mathsf{NKG}}(\mathsf{SK}'_2), \mathsf{shk}_2).$$

Hence, applying Lemma 12, it's apparent that

$$|\Pr[\text{Game 9}] - \Pr[\text{Game 10}]| \leq q^2 \sqrt{2\epsilon_3 + \frac{1}{|\mathcal{X}|}} + \frac{q^2}{|\mathcal{W}|} \leq \mathsf{negl}(\lambda).$$

Combining all the claims together, we establish the entire proof. □

F Proof of Theorem 7

In this section, we give the full proof of Theorem 7, that we show Π_{DPKE} is indifferentiable from an ideal DPKE.

Proof. According to the definition of indifferentiability, we immediately observe that, the adversary has three honest interfaces $(\mathsf{DKG}, \mathsf{DE}, \mathsf{DD})$ and six adversarial interfaces (H_0, H_1, P, P^{-1}, $\mathsf{NKG}, \mathsf{NSK}$). Therefore, we need to build an efficient simulator $\mathcal{S}$ that can simulate the six adversarial interfaces $H_0, H_1 P, P^{-1}, \mathsf{NKG}$ and NSK properly, which means, 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, there exists a system that responds to all of the queries(both honest and adversarial) in a slightly different way and then we build our simulator $\mathcal{S}$ as the system in the last game. Before the description of the games, we first specify some parameters:

- there are nine types of query: $(\mathsf{SK}; H_0), (\mathsf{PK}, \mathsf{M}; H_1), (\mathsf{PK}, \mathsf{M}; P), (\mathsf{Z}; P^{-1}), (\mathsf{sk}, \mathsf{NKG}), (\mathsf{pk}_1, \mathsf{sk}_2; \mathsf{NSK}), (\mathsf{SK}; \mathsf{DKG}), (\mathsf{PK}, \mathsf{M}; \mathsf{DE}), (\mathsf{C}, \mathsf{SK}; \mathsf{DD})$ where $\mathsf{M} \leftarrow \mathcal{M}, (\mathsf{PK}, \mathsf{pk}_1) \leftarrow \mathcal{Y}, (\mathsf{SK}, \mathsf{sk}, \mathsf{sk}_2) \leftarrow \mathcal{X}, \mathsf{Z} \leftarrow \mathcal{Z}, \mathsf{C} \leftarrow \mathcal{C}$;
- adversary makes at most q queries to the system, where $q = \mathsf{poly}(\lambda)$;
- the oracles used in the real world are $\tilde{H}_0, \tilde{H}_1, \tilde{P}, \tilde{P}^{-1}, \widetilde{\mathsf{NKG}}, \widetilde{\mathsf{NSK}}, \widetilde{\mathsf{DKG}}, \widetilde{\mathsf{DE}}$, and $\widetilde{\mathsf{DD}}$;
- in each game, the system's responses are denoted as $H_0^{\mathsf{r}}, H_1^{\mathsf{r}}, P^{\mathsf{r}}, P^{-1\mathsf{r}}, \mathsf{NKG}^{\mathsf{r}}, \mathsf{NSK}^{\mathsf{r}}, \mathsf{DKG}^{\mathsf{r}}, \mathsf{DE}^{\mathsf{r}}$ and DD^{r}, for instance, we denote $H_0^{\mathsf{r}}(\mathsf{SK})$ as the system's response when adversary makes a query $Q = (\mathsf{SK}; H_0)$;

Next, we define the hybrid games and after each game we give a rough intuition for why that game is indistinguishable from the previous game. Specifically,

Game 0. This game is identical to the real game except that the system maintains six tables, referring to H_0-table, H_1-table, P-table, P^{-1}-table, NKG-table and NSK-table. Specifically, the system responds to the queries the same as in the real world, for instance, $H_0^{\mathsf{r}}(\mathsf{SK}) = \tilde{H}_0(\mathsf{SK}), \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{DKG}}(\mathsf{SK})$ and so forth. For the tables, the system maintains them as follows:

- H_0-table: Initially empty, consists of tuples with form of $(\mathsf{SK}, \mathsf{sk}, \mathsf{PK})$. Once the adversary makes a query $(\mathsf{SK}^*; H_0)$ which does not exist in H_0-table (no tuple that the first element of it is SK^*), then the system inserts $(\mathsf{SK}^*, \tilde{H}_0(\mathsf{SK}^*), \widetilde{\mathsf{NKG}}(\tilde{H}_0(\mathsf{SK}^*))$ into H_0-table.
- H_1-table: Initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{sk}, \mathsf{PK})$. Once the adversary makes a query $(\mathsf{PK}^*, \mathsf{M}^*; H_1)$ which does not exist in H_1-table, then inserts $(\mathsf{PK}^*, \mathsf{M}^*, \tilde{H}_1(\mathsf{PK}^*||\mathsf{M}^*), \widetilde{\mathsf{NKG}}(\tilde{H}_1(\mathsf{PK}^*||\mathsf{M}^*))$ into H_1-table.
- P-table: Initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$. Once the adversary makes a query $(\mathsf{PK}^*, \mathsf{M}^*; P)$ which does not exist in P-table, it inserts $(\mathsf{PK}^*, \mathsf{M}^*, \tilde{P}(\mathsf{PK}^*||\mathsf{M}^*))$ into P-table.
- P^{-1}-table: Initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$. Once the adversary makes a query $(\mathsf{Z}^*; P^{-1})$ which does not exist in P^{-1}-table, it inserts $(\tilde{P}^{-1}(\mathsf{Z}^*), \mathsf{Z}^*)$ into P^{-1}-table(here we abuse $\tilde{P}^{-1}(\mathsf{Z}^*)$ to be 2 elements, $(\mathsf{PK}^*, \mathsf{M}^*)$).
- NKG-table. Initially empty, consists of tuples with form of $(*, \mathsf{sk}, \mathsf{PK})$. Once the adversasry makes a query $(\mathsf{sk}^*; \mathsf{NKG})$ which does not exist in NKG-table, the system inserts $(*, \mathsf{sk}^*, \widetilde{\mathsf{NKG}}(\mathsf{sk}^*))$ into NKG table.
- NSK-table. Initially empty, consists of tuples with form $(\mathsf{pk}_1, \mathsf{sk}_2; \mathsf{NSK})$. Once the adversary makes a query $(\mathsf{pk}_1, \mathsf{sk}_2; \mathsf{NSK})$ which does not exist in NSK table, the system inserts $(\mathsf{pk}_1, \mathsf{sk}_2; \widetilde{\mathsf{NSK}}(\mathsf{pk}_1, \mathsf{sk}_2))$ into NSK table.

Note that at this point all the queries are responded by the real oracles and these tables are just keeping track of information related to adversary'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.

Same as above, we here also define a relation between query Q and the table Tab. Concretely, if Q is a H_0 query $(Q = (\mathsf{SK}; H_0))$, we say $Q \in \mathsf{Tab}$ if there is a 3-tuple $T = (T_1, T_2, T_3)$ in Tab such that $T_1 = \mathsf{SK}$. Analogously, if Q is a H_1 query $(Q = (\mathsf{PK}, \mathsf{M}; H_1))$, we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{Tab}$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$. And for a P query $Q = (\mathsf{PK}, \mathsf{M}; P)$, we say $Q \in \mathsf{Tab}$ if there is a 3-tuple $T = (T_1, T_2, T_3) \in \mathsf{Tab}$ such that such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$; if Q is a P^{-1} query $(Q = (\mathsf{Z}; P^{-1}))$, we say $Q \in \mathsf{Tab}$ if there is a 3-tuple $T = (T_1, T_2, T_3) \in sfTab$ such that $T_3 = \mathsf{Z}$. For NKG query, we say $Q = (\mathsf{sk}; \mathsf{NKG}) \in \mathsf{Tab}$ if there exists a 3-tuple $T = (T_1, T_2, T_3)$ such that $T_2 = \mathsf{sk}$; and for NSK query, by $Q = (\mathsf{pk}_1, \mathsf{sk}_2; \mathsf{NSK}) \in \mathsf{Tab}$ we mean there exists a 3-tuple $T = (T_1, T_2, T_3) \in \mathsf{Tab}$ such that $T_1 = \mathsf{pk}_1, T_2 = \mathsf{sk}_2$.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding the queries. Specifically,

H_0-QUERY. Suppose $Q_k = (\mathsf{SK}; H_0)$ $(k \in [1, q])$, then the system responds as follows:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3) \in H_0$ such that $T_1 = \mathsf{SK}$, then the system responds with T_2;

– Case 2. Otherwise, the system responds with $\tilde{H}_0(\mathsf{SK})$ and inserts $(\mathsf{SK}, \tilde{H}_0(\mathsf{SK}), \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0-table.

H_1-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}; H_1)$, then the system responds as follows:

– Case 1. If $Q_k \in H_1$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$, then the system responds with T_3;
– Case 2. Otherwise, the system responds with $\tilde{H}_0(\mathsf{PK}||\mathsf{M})$ and inserts $(\mathsf{PK}, \mathsf{M}, \tilde{H}_0(\mathsf{PK}||\mathsf{M}), \mathsf{C_1})$ into H_0-table.

NKG-QUERY. Suppose $Q_k = (\mathsf{sk}; \mathsf{NKG})$, then the system responds as follows:

– Case 1. If $Q_k \in \mathsf{NKG}$, which means there is a tuple $T = (T_1, T_2, T_3) \in H_1$ such that $T_2 = \mathsf{sk}$, then the system responds with T_3;
– Case 2. If $Q_k \notin \mathsf{NKG}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ or $T' = (T'_1, T'_2, T'_3) \in H_0$ such that $T_3 = \mathsf{sk}$ or $T'_2 = \mathsf{sk}$, then the system responds to Q_k with T_4 or T'_3, and inserts $(*, \mathsf{sk}, T_4)$ or $(*, \mathsf{sk}, T'_3)$ into NKG table.
– Case 3. Otherwise, the system responds with $\widetilde{\mathsf{NKG}}(\mathsf{sk})$ and inserts $(*, \mathsf{sk}, \widetilde{\mathsf{NKG}}(\mathsf{sk}))$ into NKG-table.

P-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}, P)$, then the system responds as follows:

– Case 1. If $Q_k \in P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3) \in P \cup P^{-1}$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$, then the system responds with T_3;
– Case 2. If $Q_k \notin P \cup P^{-1}$, then the system makes a query $(\mathsf{PK}, \mathsf{M}; \mathsf{DE})((\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_4 = \mathsf{C_1}, T'_1 = \mathsf{PK}, T'_2 = T_3$, then the system responds to Q_k with $\mathsf{C_2} \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C_2} \oplus T'_3)$ into NSK table.
– Case 3. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_3 = \mathsf{C_1}, T'_1 = \mathsf{PK}, T'_2 = T_2$, then the system responds to Q_k with $\mathsf{C_2} \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C_2} \oplus T'_3)$ into NSK table.
– Case 4. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_4 = \mathsf{PK}, T'_1 = \mathsf{C_1}, T'_2 = T_3$, then the system responds to Q_k with $\mathsf{C_2} \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C_2} \oplus T'_3)$ into NSK table.
– Case 5. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_3 = \mathsf{PK}, T'_1 = \mathsf{C_1}, T'_2 = T_2$, then the system responds to Q_k with $\mathsf{C_2} \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C_2} \oplus T'_3)$ into NSK table.
– Case 6. Otherwise, the system responds with $\tilde{P}(\mathsf{PK}, \mathsf{M})$.

NSK-QUERY. Suppose $(\mathsf{pk_1}, \mathsf{sk_2}, \mathsf{NSK})$ is the k-th query, the system responds as follows:

– Case 1. If $Q_k \in \mathsf{NSK}$, which means there exists $T = (T_1, T_2, T_3) \in \mathsf{NSK}$ such that $T_1 = \mathsf{pk_1}, T_2 = \mathsf{sk_2}$, then the system responds with T_3;

- Case 2. If there exists three tuples $T = (T_1, T_2, T_3), T' = (T'_1, T'_2, T'_3) \in H_0 \cup \mathsf{NKG}, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk}_1, T'_2 = \mathsf{sk}_2, \bar{T}_1 = T'_3$ and $\bar{T}_2 = T_2$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, \bar{T}_3)$ into NSK table.
- Case 3. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk}_1, T'_3 = \mathsf{sk}_2, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_2$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, \bar{T}_3)$ into NSK table.
- Case 4. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_2 = \mathsf{sk}_2, T'_4 = \mathsf{pk}_1, \bar{T}_1 = T_3$ and $\bar{T}_2 = T'_3$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, \bar{T}_3)$ into NSK table.
- Case 5. If there exists three tuples $T = (T_1, T_2, T_3, T_4), T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_4 = \mathsf{pk}_1, T'_3 = \mathsf{sk}_2, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_3$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, \bar{T}_3)$ into NSK table.
- Case 6. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T_1 = T'_1 = \mathsf{pk}_1, T_2 = T'_2$ and $T_3 = \mathsf{sk}_2$, then the system makes a query $(T_1, T_2; \mathsf{DE})$, responds to Q_k with $\mathsf{C}_2 \oplus T'_3$, and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, T'_3)$ into NSK table.
- Case 7. If there exist two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T'_1 = T_3, T_2 = \mathsf{sk}_2$ and $\mathsf{C}_1 = \mathsf{pk}_1$, where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(T'_1, T'_2)$, then the system responds to Q_k with $\mathsf{C}_2 \oplus T'_3$ and inserts $(\mathsf{pk}_1, \mathsf{sk}_2, T'_3)$ into NSK table.
- Case 8. Otherwise, the system responds to Q_k with $\widetilde{\mathsf{NSK}}(\mathsf{pk}_1, \mathsf{sk}_2)$.

P^{-1} QUERY. Suppose $Q_k = (\mathsf{Z}, P^{-1})$, then the system responds as follows:

- Case 1. If $Q_k \in P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3) \in P \cup P^{-1}$ such that $\mathsf{T_3} = \mathsf{Z}$, then the system responds to Q_k with (T_1, T_2);
- Case 2. If $Q_k \notin P \cup P^{-1}$ but there are two tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 = T'_1, T_3 = T_2$ and $\mathsf{Z} = \mathsf{C}_2 \oplus T'_3$, where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(T_1, T_2)$, then the system responds to Q_k with (T_1, T_2), and inserts (T_1, T_2, Z) into P^{-1} table;
- Case 3. If there are two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 \neq *, T_2 = T'_2$ and $\mathsf{DD}^{\mathsf{r}}((T'_1, \mathsf{Z} \oplus T'_3), T_1) = \mathsf{M} \neq \perp$, then the system responds to Q_k with (T_3, M) and inserts $(T_3, \mathsf{M}, \mathsf{Z})$ into P^{-1} table;
- Case 4. Otherwise, the system responds with $\tilde{P}^{-1}(\mathsf{Z})$.

Note that, in Game 1 the system keeps a longer table, and for part of the queries, the system 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 correspond to the response of queries that are answered by the real oracles, and for any $\mathsf{SK}, \mathsf{PK}, \mathsf{M}$ and any ciphertext C, we have $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{DKG}}(\mathsf{SK}), \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}) = \widetilde{\mathsf{DE}}(\mathsf{PK}, \mathsf{M})$ and $\mathsf{DD}^{\mathsf{r}}(\mathsf{C}, \mathsf{SK}) = \widetilde{\mathsf{DD}}(\mathsf{C}, \mathsf{SK})$. 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 system 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 H_0 queries. Suppose $Q_k = (\mathsf{SK}; H_0)$, then the system responds:

- Case 1. If $Q_k \in H_0$, same as in Game 1;
- Case 2. Otherwise, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with sk and inserts $(\mathsf{SK}, \mathsf{sk}, \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0 table.

The only difference between Game 1 and Game 2 occurs in the case 2, where $Q_k \notin H_0$ and $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$ never appears in (NKG) table. In Game 1, the system responds with $\tilde{H}_0(\mathsf{SK})$ while in Game 2, we replace it with a random string. Due to definition, we note that the only case that the system makes a query $(\mathsf{SK}; H_0)$ is when the adversary knows nothing of $\tilde{H}_0(\mathsf{SK})$, although the adversary might know $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$. Therefore, from the adversary's view, $\tilde{H}_0(\mathsf{SK})$ is uniformly distributed in $\mathcal{X}$, which implies $\mathcal{A}$'s view preserves whp. if the system responds to Q_k with sk and implicitly set $\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}) = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$ afterwards.

Game 3. This game is identical to Game 2, except for responding to H_1 queries. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}; H_1)$, then the system responds:

- Case 1. If $Q_k \in H_1$, same as in Game 2;
- Case 2. Otherwise, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with sk and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{sk}, \mathsf{C}_1)$ into H_1 table.

The only difference between Game 2 and Game 3 occurs in the case 2, where $Q_k \notin H_1$ and C_1 never appears in (NKG)table. In Game 2, the system responds with $\tilde{H}_1(\mathsf{PK}, \mathsf{M})$ while in Game 3, we replace it with a random string. Due to definition, we note that the only case that the system makes a query $(\mathsf{PK}, \mathsf{M}; H_1)$ is when the adversary knows nothing of $\tilde{H}_1(\mathsf{PK}, \mathsf{M})$, although the adversary might know C_1. Therefore, from the adversary's view, $\tilde{H}_1(\mathsf{PK}, \mathsf{M})$ is uniformly distributed in $\mathcal{X}$, which implies $\mathcal{A}$'s view preserves whp. if the system responds to Q_k with sk and implicitly set $\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}) = \mathsf{C}_1$ afterwards.

Game 4. This game is identical to Game 2, except for responding to NKG queries. Suppose $Q_k = (\mathsf{sk}; \mathsf{NKG})$, then the system responds:

- Case 1. If $Q_k \in \mathsf{NKG}$, same as in Game 3;
- Case 2. If $Q_k \notin \mathsf{NKG}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ or $T' = (T'_1, T'_2, T'_3) \in H_0$ such that $T_3 = \mathsf{sk}$ or $T'_2 = \mathsf{sk}$, then same as in Game 3;
- Case 3. Otherwise, the system samples $\mathsf{SK}^* \xleftarrow{\$} \mathcal{X}$, responds to Q_k with $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$, and inserts $(\mathsf{SK}^*, \mathsf{sk}, \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*))$ into NKG table.

The only difference between Game 3 and Game 4 occurs in the case 3, where $Q_k \notin \mathsf{NKG}$ and sk never appears in $H_0 \cup H_1$ table. In Game 3, the system responds with $\widetilde{\mathsf{NKG}}(\mathsf{sk})$, while in Game 4, the system replaces it with a random public key($\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{SK} \xleftarrow{\$} \mathcal{X}$). We immediately note that SK is independent of $\mathcal{A}$'s view, which implies that $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$ is well-distributed.

The reason why the response is a random public key, rather than a random string, is to keep the consistency conditions. If not, the adversary could do the following attack: 1)$\mathcal{A}$ picks a secret key sk^* and makes a query $(\mathsf{sk}^*; \mathsf{NKG})(\mathsf{pk}^* = \mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*))$; 2) $\mathcal{A}$ picks a message M and makes a query $(\mathsf{pk}^*, \mathsf{M}; \mathsf{DE})((\mathsf{C}_1^*, \mathsf{C}_2^*) = \mathsf{DE}^{\mathsf{r}}(\mathsf{pk}^*, \mathsf{M}))$; 3) $\mathcal{A}$ makes a query $(\mathsf{C}_1^*, \mathsf{sk}^*; \mathsf{NKG})$; 4) $\mathcal{A}$ makes a query $(C_2^* \oplus \mathsf{NKG}^{\mathsf{r}}(\mathsf{C}_1^*, \mathsf{sk}^*); P^{-1})$ and test if the response is identical to $(\mathsf{pk}^*, \mathsf{M})$. We note that, from the attack above, the message M is independent of the system(calling $\tilde{P}^{-1}$ would not help as $\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*)$ is a random string now). Therefore the only case to extract M is to call the decryption oracle DD, which means the system has to know the corresponding secret key.

Game 5. This game is identical to Game 4, except for responding to P queries. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}; P)$, then the system responds:

- Case 1. If $Q_k \in P \cup P^{-1}$, same as in Game 4;
- Case 2. If $Q_k \notin P \cup P^{-1}$, then the system makes a query $(\mathsf{PK}, \mathsf{M}; \mathsf{DE})((\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T_1', T_2', T_3') \in \mathsf{NSK}$ such that $T_4 = \mathsf{C}_1, T_1' = \mathsf{PK}, T_2' = T_3$, then same as in Game 4;
- Case 3. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T_1', T_2', T_3') \in \mathsf{NSK}$ such that $T_3 = \mathsf{C}_1, T_1' = \mathsf{PK}, T_2' = T_2$, then same as in Game 4;
- Case 4. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T_1', T_2', T_3') \in \mathsf{NSK}$ such that $T_4 = \mathsf{PK}, T_1' = \mathsf{C}_1, T_2' = T_3$, then same as in Game 4;
- Case 5. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T_1', T_2', T_3') \in \mathsf{NSK}$ such that $T_3 = \mathsf{PK}, T_1' = \mathsf{C}_1, T_2' = T_2$, then same as in Game 4;
- Case 6. Otherwise, the system samples $\mathsf{Z} \xleftarrow{\$} \mathcal{Z}$, responds to Q_k with Z and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$ into P table.

The only difference between Game 5 and Game 4 occurs in case 6; in Game 4 the system responds with $\tilde{P}(\mathsf{PK}, \mathsf{M})$ while in Game 5 the system replaces it with a random string. By definition, we note that the only case that the system makes a query $(\mathsf{PK}, \mathsf{M}; P)$ is when adversary knows nothing of $\tilde{P}(\mathsf{PK}, \mathsf{M})$, however it might knows $\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$ and $\tilde{P}(\mathsf{PK}, \mathsf{M})$ is part of them. If the adversary also knows $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$ or $\mathsf{NSK}^{\mathsf{r}}(\mathsf{C}_1, H_0^{\mathsf{r}}(\mathsf{SK}))$, then the system must responds to Q_k correctly, and in fact this situation is covered by case 2, 3, 4, 5. Therefore, in case 6, the adversary also knows nothing of $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$ or $\mathsf{NSK}^{\mathsf{r}}(\mathsf{C}_1, H_0^{\mathsf{r}}(\mathsf{SK}))$, which implies that the system can answer it randomly.

Game 6. This game is identical to Game 5, except for responding to NSK queries. Suppose $Q_k = (\mathsf{pk}_1, \mathsf{sk}_2; \mathsf{NSK})$, then the system responds:

- Case 1. If $Q_k \in \mathsf{NSK}$, then same as in Game 5;
- Case 2. If there exists three tuples $T = (T_1, T_2, T_3), T' = (T'_1, T'_2, T'_3) \in H_0 \cup \mathsf{NKG}, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk_1}, T'_2 = \mathsf{sk_2}, \bar{T}_1 = T'_3$ and $\bar{T}_2 = T_2$, then same as in Game 5;
- Case 3. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk_1}, T'_3 = \mathsf{sk_2}, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_2$, then same as in Game 5;
- Case 4. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_2 = \mathsf{sk_2}, T'_4 = \mathsf{pk_1}, \bar{T}_1 = T_3$ and $\bar{T}_2 = T'_3$, then same as in Game 5;
- Case 5. If there exists three tuples $T = (T_1, T_2, T_3, T_4), T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_4 = \mathsf{pk_1}, T'_3 = \mathsf{sk_2}, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_3$, then same as in Game 5;
- Case 6. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T_1 = T'_1 = \mathsf{pk_1}, T_2 = T'_2$ and $T_3 = \mathsf{sk_2}$, then same as in Game 5;
- Case 7. If there exist two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T'_1 = T_3, T_2 = \mathsf{sk_2}$ and $\mathsf{C_1} = \mathsf{pk_1}$, where $(\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(T'_1, T'_2)$, then same as in Game 5;
- Case 8. Otherwise, the system samples $\mathsf{Z} \xleftarrow{\$} \mathcal{Z}$, responds to Q_k with Z and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \mathsf{Z})$ into NSK table.

The only difference between Game 6 and Game 5 occurs in case 8; in Game 5 the system responds with $\widetilde{\mathsf{NSK}}(\mathsf{pk_1}, \mathsf{sk_2})$ while in Game 6 the system replace it with a random string. By definition, we note that, by the consistency conditions, there are three cases in total to predict $\mathsf{NSK}^r(\mathsf{pk_1}, \mathsf{sk_2})$. Case A: the adversary knows $\mathsf{NSK}^r(\mathsf{pk_2}, \mathsf{sk_1})$ where $\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}_\ell) = \mathsf{pk}_\ell$(Case A is covered by case 2, 3, 4, 5); Case B: following the encryption path, the adversary knows $P^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$ such that $\mathsf{PK} = \mathsf{pk_1}$ and $H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}) = \mathsf{sk_2}$(covered by case 6); Case C: following the decryption path, the adversary knows $P^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$ such that $\mathsf{pk_1} = \mathsf{C_1}, \mathsf{PK} = \mathsf{NKG}^{\mathsf{r}}(\mathsf{sk_2})$ where $(\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$(covered by case 7).Therefore, in case 8, the adversary cannot predict $\mathsf{NSK}^{\mathsf{r}}(\mathsf{pk_1}, \mathsf{sk_2})$ at all, which implies that the system can answer it randomly.

Game 7. This game is identical to Game 5, except for responding to P^{-1} queries. Suppose $Q_k = (\mathsf{Z}; P^{-1})$, then the system responds: P^{-1} QUERY. Suppose $Q_k = (\mathsf{Z}, P^{-1})$, then the system responds as follows:

- Case 1. If $Q_k \in P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3) \in P \cup P^{-1}$ such that $\mathsf{T_3} = \mathsf{Z}$, then same as in Game 6;
- Case 2. If $Q_k \notin P \cup P^{-1}$ but there are two tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 = T'_1, T_3 = T_2$ and $\mathsf{Z} = \mathsf{C_2} \oplus T'_3$, where $(\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(T_1, T_2)$, then same as in Game 6;
- Case 3. If there are two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 \neq *, T_2 = T'_2$ and $\mathsf{DD}^{\mathsf{r}}((T'_1, \mathsf{Z} \oplus T'_3), T_1) = \mathsf{M} \neq \perp$, then same as in Game 6;

– Case 4. Otherwise, the system samples $\mathsf{PK} \xleftarrow{\$} \mathcal{Y}, \mathsf{M} \xleftarrow{\$} \mathcal{M}$, responds with $(\mathsf{PK}, \mathsf{M})$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$ into P^{-1} table.

The only difference between Game 7 and Game 6 occurs in case 4; in Game 6 the system responds with $\tilde{P}^{-1}(\mathsf{Z})$ while in Game 7 the system replace it with a random string. Same as above, we note that there are two cases in total that the adversary can predicate $P^{-1\mathsf{r}}(\mathsf{Z})$ before Q_k, which are exactly case 2 and case 3(following the encryption and decryption path, respectively). Hence, in case 4, it's proper for the system to responds to Q_k with a random string.

Game 8. In Game 7, the queries to the adversarial interfaces are answered by the tables which're maintained by the system and by making queries to $\mathsf{DKG}, \mathsf{DE}, \mathsf{DD}$. The system never makes queries directly to $\tilde{H}_0, \tilde{H}_1, \tilde{P}, \tilde{P}^{-1}, \widetilde{\mathsf{NKG}}, \widetilde{\mathsf{NSK}}$; these oracles are *only* used to answer the $\mathsf{DKG}, \mathsf{DE}, \mathsf{DD}$ queries (either generated by the adversary or by the system's response to $H_0, H_1, P, P^{-1}, \mathsf{NKG}, \mathsf{NSK}$ queries). At this point, it is straightforward to show that we can replace $\mathsf{DKG}, \mathsf{DE}$ and DD with the ideal versions from Definition 6, resulting in Game 8.

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

Simulator In Ideal Game. Let $(\mathsf{KEYGEN}, \mathsf{ENC}, \mathsf{DEC})$ be the function pair that samples from $\mathcal{T}_{\mathsf{DPKE}}$, the simulator works as follows. Like the system in Game 8, the simulator also maintains six tables, referring to H_0-table, H_1-table, P-table, P^{-1}-table, NKG-table and NSK-table. Concretely:

– H_0-table: initially empty, consists of tuples with form of $(\mathsf{SK}, \mathsf{sk}, \mathsf{PK})$;
– H_1-table: initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{sk}, \mathsf{PK})$;
– P-table: initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$;
– P^{-1}-table: initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$;
– NKG-table: initially empty, consists of tuples with form of $(\mathsf{SK}, \mathsf{sk}, \mathsf{PK})$;
– NSK-table: initialy empty, consists of tuples with form of $(\mathsf{pk}_1, \mathsf{sk}_2, \mathsf{Z})$.

By definition, the simulator $\mathcal{S}$ has access to the honest interfaces $(\mathsf{DKG}, \mathsf{DD}, \mathsf{DE})$, where $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{KEYGEN}(\mathsf{SK}), \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}) = \mathsf{ENC}(\mathsf{PK}, \mathsf{M}), \mathsf{DD}^{\mathsf{r}}(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK}) = \mathsf{DEC}(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK})$. And for the adversarial queries, $\mathcal{S}$ works as the system in Game 8, by just using the tables and calling the honest interfaces.

H_0-QUERY. Suppose $Q_k = (\mathsf{SK}; H_0)$ $(k \in [1, q])$, then the system responds as follows:

– Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3) \in H_0$ such that $T_1 = \mathsf{SK}$, then the system responds with T_2;
– Case 2. Otherwise, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with sk and inserts $(\mathsf{SK}, \mathsf{sk}, \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}))$ into H_0 table.

H_1-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}; H_1)$, then the system responds as follows:

- Case 1. If $Q_k \in H_1$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$, then the system responds with T_3;
- Case 2. Otherwise, the system samples $\mathsf{sk} \xleftarrow{\$} \mathcal{X}$, responds to Q_k with sk and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{sk}, \mathsf{C}_1)$ into H_1 table.

NKG-QUERY. Suppose $Q_k = (\mathsf{sk}; \mathsf{NKG})$, then the system responds as follows:

- Case 1. If $Q_k \in \mathsf{NKG}$, which means there is a tuple $T = (T_1, T_2, T_3) \in H_1$ such that $T_2 = \mathsf{sk}$, then the system responds with T_3;
- Case 2. If $Q_k \notin \mathsf{NKG}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_1$ or $T' = (T'_1, T'_2, T'_3) \in H_0$ such that $T_3 = \mathsf{sk}$ or $T'_2 = \mathsf{sk}$, then the system responds to Q_k with T_4 or T'_3, and inserts $(*, \mathsf{sk}, T_4)$ or $(*, \mathsf{sk}, T'_3)$ into NKG table.
- Case 3. Otherwise, the system samples $\mathsf{SK}^* \xleftarrow{\$} \mathcal{X}$, responds to Q_k with $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$, and inserts $(\mathsf{SK}^*, \mathsf{sk}, \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*))$ into NKG table.

P-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}, P)$, then the system responds as follows:

- Case 1. If $Q_k \in P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3) \in P \cup P^{-1}$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$, then the system responds with T_3;
- Case 2. If $Q_k \notin P \cup P^{-1}$, then the system makes a query $(\mathsf{PK}, \mathsf{M}; \mathsf{DE})((\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$). If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_4 = \mathsf{C}_1, T'_1 = \mathsf{PK}, T'_2 = T_3$, then the system responds to Q_k with $\mathsf{C}_2 \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C}_2 \oplus T'_3)$ into NSK table.
- Case 3. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_3 = \mathsf{C}_1, T'_1 = \mathsf{PK}, T'_2 = T_2$, then the system responds to Q_k with $\mathsf{C}_2 \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C}_2 \oplus T'_3)$ into NSK table.
- Case 4. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_4 = \mathsf{PK}, T'_1 = \mathsf{C}_1, T'_2 = T_3$, then the system responds to Q_k with $\mathsf{C}_2 \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C}_2 \oplus T'_3)$ into NSK table.
- Case 5. Similarly, if there exist tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_3 = \mathsf{PK}, T'_1 = \mathsf{C}_1, T'_2 = T_2$, then the system responds to Q_k with $\mathsf{C}_2 \oplus T'_3$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{C}_2 \oplus T'_3)$ into NSK table.
- Case 6. Otherwise, the system samples $\mathsf{Z} \xleftarrow{\$} \mathcal{Z}$, responds to Q_k with Z and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$ into P table.

NSK-QUERY. Suppose $(\mathsf{pk}_1, \mathsf{sk}_2, \mathsf{NSK})$ is the k-th query, the system responds as follows:

- Case 1. If $Q_k \in \mathsf{NSK}$, which means there exists $T = (T_1, T_2, T_3) \in \mathsf{NSK}$ such that $T_1 = \mathsf{pk}_1, T_2 = \mathsf{sk}_2$, then the system responds with T_3;

- Case 2. If there exists three tuples $T = (T_1, T_2, T_3), T' = (T'_1, T'_2, T'_3) \in H_0 \cup \mathsf{NKG}, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk_1}, T'_2 = \mathsf{sk_2}, \bar{T}_1 = T'_3$ and $\bar{T}_2 = T_2$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \bar{T}_3)$ into NSK table.
- Case 3. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_3 = \mathsf{pk_1}, T'_3 = \mathsf{sk_2}, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_2$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \bar{T}_3)$ into NSK table.
- Case 4. If there exists three tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}, T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_2 = \mathsf{sk_2}, T'_4 = \mathsf{pk_1}, \bar{T}_1 = T_3$ and $\bar{T}_2 = T'_3$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \bar{T}_3)$ into NSK table.
- Case 5. If there exists three tuples $T = (T_1, T_2, T_3, T_4), T' = (T'_1, T'_2, T'_3, T'_4) \in H_1, \bar{T} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$ such that $T_4 = \mathsf{pk_1}, T'_3 = \mathsf{sk_2}, \bar{T}_1 = T'_4$ and $\bar{T}_2 = T_3$, then the system responds to Q_k with $\bar{T}_3$ and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \bar{T}_3)$ into NSK table.
- Case 6. If there exist tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T_1 = T'_1 = \mathsf{pk_1}, T_2 = T'_2$ and $T_3 = \mathsf{sk_2}$, then the system makes a query $(T_1, T_2; \mathsf{DE})$, responds to Q_k with $\mathsf{C_2} \oplus T'_3$, and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, T'_3)$ into NSK table.
- Case 7. If there exist two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in P \cup P^{-1}$ such that $T'_1 = T_3, T_2 = \mathsf{sk_2}$ and $\mathsf{C_1} = \mathsf{pk_1}$, where $(\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(T'_1, T'_2)$, then the system responds to Q_k with $\mathsf{C_2} \oplus T'_3$ and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, T'_3)$ into NSK table.
- Case 8. Otherwise, the system samples $\mathsf{Z} \xleftarrow{\$} \mathcal{Z}$, responds to Q_k with Z and inserts $(\mathsf{pk_1}, \mathsf{sk_2}, \mathsf{Z})$ into NSK table.

P^{-1} QUERY. Suppose $Q_k = (\mathsf{Z}, P^{-1})$, then the system responds as follows:

- Case 1. If $Q_k \in P \cup P^{-1}$, which means there is a tuple $T = (T_1, T_2, T_3) \in P \cup P^{-1}$ such that $\mathsf{T_3} = \mathsf{Z}$, then the system responds to Q_k with (T_1, T_2);
- Case 2. If $Q_k \notin P \cup P^{-1}$ but there are two tuples $T = (T_1, T_2, T_3, T_4) \in H_1$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 = T'_1, T_3 = T_2$ and $\mathsf{Z} = \mathsf{C_2} \oplus T'_3$, where $(\mathsf{C_1}, \mathsf{C_2}) = \mathsf{DE}^{\mathsf{r}}(T_1, T_2)$, then the system responds to Q_k with (T_1, T_2), and inserts (T_1, T_2, Z) into P^{-1} table;
- Case 3. If there are two tuples $T = (T_1, T_2, T_3) \in H_0 \cup \mathsf{NKG}$ and $T' = (T'_1, T'_2, T'_3) \in \mathsf{NSK}$ such that $T_1 \neq *, T_2 = T'_2$ and $\mathsf{DD}^{\mathsf{r}}((T'_1, \mathsf{Z} \oplus T'_3), T_1) = \mathsf{M} \neq \perp$, then the system responds to Q_k with (T_3, M) and inserts $(T_3, \mathsf{M}, \mathsf{Z})$ into P^{-1} table;
- Case 4. Otherwise, the system samples $\mathsf{PK} \xleftarrow{\$} \mathcal{Y}, \mathsf{M} \xleftarrow{\$} \mathcal{M}$, responds with $(\mathsf{PK}, \mathsf{M})$ and inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{Z})$ into P^{-1} table.

Now we are ready to prove the indistinguishability between any adjacent games.

Claim. Game Real $\approx$ Game 0.

Proof. The only difference between Game Real and Game 0 is that, in Game 0 the system additionally maintains several tables that are completely hidden

from the adversary, hence we have

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

Claim. Game 0 $\approx$ Game 1.

Proof. By definition, we note that in Game 1, the system maintains longer tables and it responds to part of the queries (type A query) by just using those tables and calling the honest interfaces. For the other queries (type B query), the system responds by calling the real oracles. Moreover, the items stored in those tables are always consistent with the real oracles, and in either games, the honest interfaces correspond to the real oracles, which means the response by calling the honest interfaces is identical to the one by calling real oracles(for instance, $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{DKG}}(\mathsf{SK}) = \widetilde{\mathsf{NKG}}(\tilde{H}_0(\mathsf{SK}))$). Hence the responses of type A queries by either the real oracles(Game 0) or by honest interfaces plus tables(Game 1) are identical, which refers to

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

Claim. Game 1 $\approx$ Game 2.

Proof. Recalling that the only difference between Game 1 and Game 2 occurs in case 2, where $Q_k = (\mathsf{SK}^*; H_0) \notin H_0$ and $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ never appears in the previous queries. In Game 1, the system responds to Q_k with $\tilde{H}_0(\mathsf{PK}^*)$ while in Game 2, the system replaces it with a random string sk^* in $\mathcal{X}$. To prove the indistinguishability, we first formalize the adversary's view in Game 1. By definition, we immediately observe that, in Game 1,

- The system responds to (SK, H_0) with $\tilde{H}_0(\mathsf{SK})$;
- The system responds to $(\mathsf{PK}, \mathsf{M}; P)$ with $\tilde{P}(\mathsf{PK}, \mathsf{M})$;
- The system responds to $(\mathsf{Z}; P^{-1})$ with $\tilde{P}^{-1}(\mathsf{Z})$;
- The system responds to $(\mathsf{PK}, \mathsf{M}; H_1)$ with $\tilde{H}_1(\mathsf{PK}, \mathsf{M})$;
- The system responds to $(\mathsf{sk}; \mathsf{NKG})$ with $\widetilde{\mathsf{NKG}}(\mathsf{sk})$;
- The system responds to $(\mathsf{pk}_1, \mathsf{sk}_2, \mathsf{NSK})$ with $\widetilde{\mathsf{NSK}}(\mathsf{pk}_1, \mathsf{sk}_2)$;
- The system responds to $(\mathsf{SK}; \mathsf{DKG})$ with $\widetilde{\mathsf{DKG}}(\mathsf{SK})$;
- The system responds to $(\mathsf{PK}, \mathsf{M}; \mathsf{DE})$ with $\widetilde{\mathsf{DE}}(\mathsf{PK}, \mathsf{M})$;
- The system responds to $(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK}; \mathsf{DD})$ with $\widetilde{\mathsf{DD}}(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK})$.

Hence in adversary's perspective, under the consistency conditions listed below, the responses of H_0 and H_1 are independent and random strings; the responses of NKG and NSK are random injections; the responses of P and P^{-1} are random permutations where P^{-1} is P's inverse. Here is the consistency conditions:

- $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{Z})) = \mathsf{Z}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{Z})) = \mathsf{Z}$;
- There exists no $\mathsf{SK}_1, \mathsf{SK}_2$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1) = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)$;
- There exists no $\mathsf{Z}_1, \mathsf{Z}_2$ such that $P^{\mathsf{r}}(\mathsf{Z}_1) = P^{\mathsf{r}}(\mathsf{Z}_2)$ or $P^{-1\mathsf{r}}(\mathsf{Z}_1) = P^{-1\mathsf{r}}(\mathsf{Z}_2)$;

- $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{NKG}^{\mathsf{r}}(H_0^{\mathsf{r}}(\mathsf{SK}^*))$;
- $\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}) = \mathsf{NKG}^{\mathsf{r}}(H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})), \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})) \oplus P^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$;
- $\mathsf{DD}^{\mathsf{r}}(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{C}_1, H_0^{\mathsf{r}}(\mathsf{SK})))$ if the ciphertext is valid by re-encryption;
- $\mathsf{NSK}^{\mathsf{r}}(\mathsf{pk}_1, \mathsf{sk}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{pk}_2, \mathsf{sk}_1)$ if and only if $\mathsf{pk}_\ell = \mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}_\ell)$;
- $P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M}) = \mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}_1)$ where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$;
- $P^{-1\mathsf{r}}(\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}_1)) = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$ where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$.

Now we turn to Game 2, where the system responds to Q_k with a random string sk^* in $\mathcal{X}$, comparing to $\tilde{H}_0(\mathsf{SK}^*)$ in Game 1. Next we see the view on the adversarial interfaces. For H_0 queries, as sk^* is uniformly sampled, the responses are well distributed. For H_1 queries, the responses are identical in either game except that there exists $(\mathsf{PK}, \mathsf{M})$ such that the ciphertext$\mathsf{C}_1 = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$. As $\tilde{H}_1$ is a random oracle, the probability of this bad event is bounded by $\frac{q}{|\mathcal{X}|}$. For NKG queries, note that after Q_k, the system implicitly sets $\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*) = \mathsf{DE}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{NKG}^{\mathsf{r}}(\tilde{H}_0(\mathsf{SK}^*))$. Hence as long as the adversary never makes a query $(\mathsf{sk}^*; \mathsf{NKG})$ before Q_k and $\tilde{H}_0(\mathsf{SK}^*)$ is hidden, the responses are consistent. In fact sk^* is uniformly sampled and for the adversary, the only way to have $\tilde{H}_0(\mathsf{SK}^*)$ is randomly guessing, which means the probability of the bad events is bounded by $\frac{k-1+q}{|\mathcal{X}|}$. For NSK and P queries, note that after Q_k the system implicitly sets

$$\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M})) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1);$$
$$P^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}) = \mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1),$$

where $\mathsf{PK}^* = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ and $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}(\mathsf{PK}^*, \mathsf{M})$. Note that there are two cases: 1) adversary calls $(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}; \mathsf{NSK})$ or $(\mathsf{PK}^*, \mathsf{M}; P)$ before it makes a query $(\mathsf{sk}, \mathsf{C}_1; \mathsf{NSK})$; 2) adversary calls $(\mathsf{sk}, \mathsf{C}_1; \mathsf{NSK})$ before it makes queries $(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}; \mathsf{NSK})$ and $(\mathsf{PK}^*, \mathsf{M}; P)$. For the first case, we immediately observe that the system switch the response of $(\mathsf{sk}^*, \mathsf{C}_1; \mathsf{NSK})$ to $\widetilde{\mathsf{NSK}}(\tilde{H}_0(\mathsf{SK}^*), \mathsf{C}_1)$, and under the condition $\tilde{H}_0(\mathsf{SK}^*)$ is independent of adversary's view, the response is well-distributed. Analogously, for the second case, the response of $(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}; \mathsf{NSK})$ or $(\mathsf{PK}^*, \mathsf{M}; P)$ will be set as $\mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1) = \widetilde{\mathsf{NSK}}(\mathsf{sk}^*, \mathsf{C}_1)$, due to sk^* is sampled by the adversary and $\widetilde{\mathsf{NSK}}$ is a random injection(with a shared key property), the response is well-distributed. For P^{-1} query, after Q_k, the system sets

$$P^{-1\mathsf{r}}(\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1)) = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}).$$

We observe that the adversary can not make such a query before $(\mathsf{sk}^*, \mathsf{C}_1; \mathsf{NSK})$ or $(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}; \mathsf{NSK})$. Same reason as above($\mathsf{NSK}$ or P queries), the responses are well distributed.

Next we prove that, with high probability, the consistency condition.

First equation. Under the condition that the responses of P and P^{-1} are consistent, it's trivial that the first equation holds.

Second equation. As $\widetilde{\mathsf{NKG}}$ is a random injection, it's trivial that if the adversary never makes a query $(\tilde{H}_0(\mathsf{SK}^*); \mathsf{NKG})$, the equation holds.

Third equation. For one direction, the adversary attempts to outputs $(\mathsf{PK}_1, \mathsf{M}_1) \neq (\mathsf{PK}_2, \mathsf{M}_2)$ such that $P^{\mathsf{r}}((\mathsf{PK}_1, \mathsf{M}_1)) = \mathsf{P}^{\mathsf{r}}((\mathsf{PK}_2, \mathsf{M}_2))$. While, under condition that the response is consistent, we have that

$$P^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{M}_1) = \mathsf{C}_2^1 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{H}_1^{\mathsf{r}}(\mathsf{PK}_1, \mathsf{M}_1))$$
$$P^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{M}_2) = \mathsf{C}_2^2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{H}_1^{\mathsf{r}}(\mathsf{PK}_2, \mathsf{M}_2))$$

where $(\mathsf{C}_1^\ell, \mathsf{C}_2^\ell) = \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}_\ell, \mathsf{M}_\ell)$. Therefore the response is ciphertext-related, which means with high probability, there is no collusion.

For the other direction, we note that the response is also ciphertext-related, so analogously this equation holds with high probability.

Fourth equation. If the responses of NKG are consistent, then this equation holds trivially.

Fifth equation. Under the condition that the responses of H_1 queries do not change in Game 2 and the responses of other queries are consistent, this equation holds trivially.

Sixth equation. If the responses of $\mathsf{P}^{-1}, \mathsf{NSK}$ queries are consistent, then this equation holds trivially.

Seventh equation. Note that if $\tilde{H}_0(\mathsf{SK}^*)$ is independent of the adversary's view(if not the adversary can makes a query $(\mathsf{PK}, \tilde{H}_0(\mathsf{SK}^*); \mathsf{NSK})$ and detect a collision here) and the responses are consistent, this equation holds.

Eighth equation. Analogously, we note that if $\tilde{H}_0(\mathsf{SK}^*)$ is independent of the adversary's view and the responses of all queries are consistent, this equation holds.

Ninth equation. This equation is implied by the eighth one.

Combing together, we can bound the union of the bad events as

$$\Pr[\mathsf{Bad}] \leq \frac{q}{|\mathcal{X}|} + \frac{k-1}{|\mathcal{X}|} + \frac{q}{|\mathcal{X}|} \leq \mathsf{negl}(\lambda)$$

which refers to

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

Claim. Game 2 $\approx$ Game 3.

Proof. Easy to note that the analysis in this claim is symmetrical to the last one(Game 1 $\approx$ Game 2), hence

$$|\Pr[\text{Game 2}] - \Pr[\text{Game 3}]| \leq |\Pr[\text{Game 1}] - \Pr[\text{Game 2}]|.$$

Claim. Game 3 $\approx$ Game 4.

Proof. Recalling that the only difference between Game 3 and Game 4 occurs in case 3, where $Q_k = (\mathsf{sk}^*; \mathsf{NKG}) \notin \mathsf{H}_0$. In Game 1, the system responds to Q_k with $\widetilde{\mathsf{NKG}}(\mathsf{sk}^*)$ while in Game 2, the system samples SK^* and replace the response with $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$.

Analogous to the analysis above(Game 1 $\approx$ Game 2), we know that under the condition that those bad events never occurs, the adversary's view on Game 2 is: the response of H_0, H_1 queries are random and independent random strings; NKG is an random injection and NSK is another random injection but with the shared key property, P is a random permutation and P^{-1} is its inverse. Beyond that, the responses also satisfy the following equations:

- $P^{\mathsf{r}}(P^{-1\mathsf{r}}(\mathsf{Z})) = \mathsf{Z}, P^{-1\mathsf{r}}(P^{\mathsf{r}}(\mathsf{Z})) = \mathsf{Z}$;
- There exists no $\mathsf{SK}_1, \mathsf{SK}_2$ such that $\mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_1) = \mathsf{NKG}^{\mathsf{r}}(\mathsf{SK}_2)$;
- There exists no $\mathsf{Z}_1, \mathsf{Z}_2$ such that $P^{\mathsf{r}}(\mathsf{Z}_1) = P^{\mathsf{r}}(\mathsf{Z}_2)$ or $P^{-1\mathsf{r}}(\mathsf{Z}_1) = P^{-1\mathsf{r}}(\mathsf{Z}_2)$;
- $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{NKG}^{\mathsf{r}}(H_0^{\mathsf{r}}(\mathsf{SK}^*))$;
- $\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}) = \mathsf{NKG}^{\mathsf{r}}(H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})), \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})) \oplus P^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})$;
- $\mathsf{DD}^{\mathsf{r}}(\mathsf{C}_1, \mathsf{C}_2, \mathsf{SK}) = P^{-1\mathsf{r}}(\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{C}_1, H_0^{\mathsf{r}}(\mathsf{SK})))$ if the ciphertext is valid by re-encryption;
- $\mathsf{NSK}^{\mathsf{r}}(\mathsf{pk}_1, \mathsf{sk}_2) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{pk}_2, \mathsf{sk}_1)$ if and only if $\mathsf{pk}_\ell = \mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}_\ell)$;
- $P^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M}) = \mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}_1)$ where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$;
- $P^{-1\mathsf{r}}(\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}_1)) = (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$ where $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}), \mathsf{M})$.

Next, we turn to case 3 in Game 4 and see the view on the adversarial interfaces. For H_0 query, after Q_k, the system implicitly sets $H_0^{\mathsf{r}}(\mathsf{SK}^*) = \mathsf{sk}^*$. Hence as long as the adversary never makes a query $(\mathsf{SK}^*; H_0)$ then the response would be consistent. However, sk^* is chosen by the adversary and it might not be well-distributed, hence it's necessary to prove that the adversary never makes such a query even after Q_k. In fact, SK^* is sampled by the system and never revealed to the adversary, which means $(\mathsf{SK}^*; H_0)$ never appears in the query sequence except with negligible probability($\leq \frac{(k-1)}{|\mathcal{X}|} + \frac{(q-k)}{|\mathcal{X}|}$). For H_1 queries, the responses are identical in either games unless there is a query $(\mathsf{PK}, \mathsf{M}; H_1)$ such that $\mathsf{C}_1 = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$, which is bound by $\frac{q}{|\mathcal{X}|}$. For NKG query, as long as SK^* and $\tilde{H}_0(\mathsf{SK}^*)$ are hidden from adversary, the responses are consistent and well-distributed. In fact, the only way for the adversary to gain SK^* and $\tilde{H}_0(\mathsf{SK}^*)$ is randomly guessing and the probability of success is of course negligible.

For NSK and P queries, note that after Q_k the system implicitly sets

$$\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}^*, H_1^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M})) = \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1);$$
$$P^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}) = \mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1),$$

where $\mathsf{PK}^* = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ and $(\mathsf{C}_1, \mathsf{C}_2) = \mathsf{DE}(\mathsf{PK}^*, \mathsf{M})$. Moreover, except for negligible probability($\leq \frac{q}{|\mathcal{X}|}$), $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ never appears before Q_k. Then applying the same analysis above(Game 1 $\approx$ Game 2), we have that the responses are consistent and well-distributed. For P^{-1} queries, analogously, the responses are proper except for negligible probability.

For the equations, observe that, the 1st, 3rd, 4th, 5th and 6th equations holds just applying the analysis above, and next we explore the rest ones.

Second equation. Note that the responses of NKG queries are random public keys, hence the only case that induces collision is that in the query sequence there exists $\mathsf{SK}_1 \neq \mathsf{SK}_2$ but $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}_1) = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}_2)$. As $\tilde{H}_0$ is a random oracle and $\widetilde{\mathsf{NKG}}$ is a random injection, it's apparent that this bad event is bounded by $\frac{q^2}{|\mathcal{X}|}$.

Seventh equation. Note that if $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ is independent of adversary's view before Q_k and all responses are consistent, then this equation holds.

Eight equation. Analogously, if $\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}^*)$ is independent of adversary's view before Q_k and all responses are consistent, then this equation holds.

Ninth equation. This equation is implied by the eighth one.

Remark. For the ninth equation, we note that there are two ways to generate a valid $\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1)$. The first one is from the encryption path, specifically the adversary makes the following queries

$$(\mathsf{sk}^*; \mathsf{NKG}), (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*; \mathsf{H}_1), (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*; \mathsf{DE}), (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{H}_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*); \mathsf{NSK})$$

then the system calculates:

$$\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), H_1^{\mathsf{r}}(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*)) = \mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1).$$

In this case, the system has information of $(\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*)$, so it can answer this query property. However, the second one is generate from the decryption path, concretely the adversary makes the following queries:

$$(\mathsf{sk}^*; \mathsf{NKG}), (\mathsf{NKG}^{\mathsf{r}}(\mathsf{sk}^*), \mathsf{M}^*; \mathsf{DE}), (\mathsf{sk}^*, \mathsf{C}_1; \mathsf{NSK}),$$

then it calculates $\mathsf{C}_2 \oplus \mathsf{NSK}^{\mathsf{r}}(\mathsf{sk}^*, \mathsf{C}_1)$ directly. We note that, in this case, the adversary never hand in the plaintext M^* to the system while the system is obligated to output the correct plaintext, although the min-entropy of plaintext might be very high. To do so, the system should have an alternative way to extract M^*, which is using the decryption oracle DD. And this explains why the response of $(\mathsf{sk}^*, \mathsf{NKG})$ must be a random public key, otherwise the system can not make use of the decryption oracle.

Combining together, we can bound the union of the bad events by

$$\Pr[\mathsf{Bad}] \leq \frac{k-1}{|\mathcal{X}|} + \frac{q-k}{|\mathcal{X}|} + \frac{q}{|\mathcal{X}|} + \frac{q}{|\mathcal{X}|},$$

which refers to

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

Claim. Game 4 $\approx$ Game 5.

Proof. Firstly, we note that, in either Game 4 or Game 5, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 4 and Game 5 is when Case 6 occurs where the system replaces the response of P queries (fails in the consistency check) with a random string str^*.

By definition, we immediately observe that the responses of H_0, H_1 and NKG queries are identical in either game, so it suffices to only analyze the rest one. Suppose $Q_k = (\mathsf{PK}^*, \mathsf{M}^*; P)$, the adversary's view on the adversarial interfaces are as follows:

View on P. When case 6 happens, the system implicitly sets $P(\mathsf{PK}^*, \mathsf{M}^*) = \mathsf{str}^*$. Hence if str^* never appears in the previous queries(bounded by $\frac{q}{|\mathcal{Y}|}$), then the view on P preserves.

View on NSK. When case 6 happens, the system implicitly sets

$$\mathsf{NSK}(\mathsf{PK}^*, H_1(\mathsf{PK}^*, \mathsf{M}^*)) = \mathsf{C}_2 \oplus \mathsf{str}^*$$

Hence, similar to the analysis above, if $\mathsf{C}_2 \oplus \mathsf{str}^*$ never appears in the previous queries, and $\tilde{P}(\mathsf{PK}^*, \mathsf{M}^*)$ and $(\mathsf{PK}, \mathsf{M})$ such that $\tilde{\mathsf{NSK}}(\mathsf{PK}, \mathsf{M}) = \mathsf{C}_2 \oplus \mathsf{str}^*$ never appear, then the view on NSK does not change. For the bad event we can bound it by $\frac{3q}{|\mathcal{Y}|}$.

View on P^{-1}. When case 6 occurs, the system implicitly sets $\tilde{P^{-1}}(\mathsf{C}_2 \oplus \mathsf{str}^*) = (\mathsf{PK}^*, \mathsf{M}^*)$. Hence, similar to the analysis above, if $\mathsf{C}_2 \oplus \mathsf{str}^*$ never appears in the previous queries, and $\tilde{P}(\mathsf{PK}^*, \mathsf{M}^*)$ and Z such that $\tilde{P^{-1}}(\mathsf{Z}) = (\mathsf{PK}^*, \mathsf{M}^*)$ never appears, then the view on P^{-1} preserves. For this bad event, we can also bound it by $\frac{3q}{|\mathcal{Y}|}$.

Now, we bound the union of these bad events as

$$\Pr[\mathsf{Bad}] \leq \frac{q}{|\mathcal{Y}|} + \frac{3q}{|\mathcal{Y}|} + \frac{3q}{|\mathcal{Y}|} \leq \mathsf{negl}(\lambda),$$

which refers to

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

Claim. Game 5 $\approx$ Game 6.

Proof. Easy to observe that the role of NSK is symmetric to the role of $P(\mathsf{C}_2 = \mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{H}_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M})) \oplus \mathsf{P}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$, hence applying the same analysis above(Game 4 $\approx$ Game 5), we have that

$$|\Pr[\text{Game 5}] - \Pr[\text{Game 6}]| \leq |\Pr[\text{Game 4}] - \Pr[\text{Game 5}]|.$$

Claim. Game 6 $\approx$ Game 7.

Proof. Firstly, we note that, in either Game 6 or Game 7, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 6 and Game 7 is when Case 4 occurs where the system replaces the response of P^{-1} queries (fails in the consistency check) with a random string.

By definition, we immediately observe that the responses of H_0, H_1 and NKG queries are identical in either game, so it suffices to only analyze the rest one. Suppose $Q_k = (\mathsf{Z}, \mathsf{P}^{-1})$, the adversary's view on the adversarial interfaces are as follows:

View on P and NSK. In both games, the responses of P queries are responded by P^{-1} table or a random string, therefore expect with collision, the view on P and NSK does not change. And this bad event can be bounded by $\frac{2q}{|\mathcal{Y}|}$.

View on P^{-1}. When case 4 occurs(the consistency check fails), then the system replaces $P^{-1\mathsf{r}}(\mathsf{X})$ with a random string.

Firstly we note that, in both games, the real oracles $\tilde{H}_0, \tilde{H}_1, \widetilde{\mathsf{NKG}}, \widetilde{\mathsf{NSK}}, \tilde{P}$ are hidden, hence except with negligible probability($\leq \frac{q}{|\mathcal{Z}|}$), the adversary can not output a tuple $(\mathsf{PK}, \mathsf{M}, \mathsf{str})$, such that $\tilde{P}(\mathsf{PK}, \mathsf{M}) = \mathsf{str}$. Hence, it's rest to show if the consistency check fails, with high probability, the view on P^{-1} does not change. In fact, the view changes means that the adversary, without knowing $\mathsf{NSK}^{\mathsf{r}}(\mathsf{PK}, H_1^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}))$, can output a Z such that $Z = \mathsf{NSK}(\mathsf{PK}, H_1(\mathsf{PK}, \mathsf{M})) \oplus C_2$, which is bounded by $\frac{q}{|\mathcal{Y}|}$. Combining together, we can bound the union of the bad events by

$$\Pr[\mathsf{Bad}] \leq \frac{2q}{|\mathcal{Y}|} + \frac{q}{|Y|},$$

which refers to,

$$|\Pr[\text{Game 6}] - \Pr[\text{Game 7}]| \leq q \cdot \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Claim. Game 7 $\approx$ Game 8.

Proof. Let (F, E, D) be an ideal deterministic encryption scheme that's sampled from $\mathcal{T}_{\mathsf{PKE}}$(Definition 6), and we note that in Game 7, the system responds all of the adversarial interfaces just using tables and honest interfaces, it never directly calls the real oracles. Besides, $\tilde{H}_0, \tilde{H}_1$ are random oracles, $(\widetilde{\mathsf{NKG}}, \widetilde{\mathsf{NSK}})$ is an ideal NIKE and $\tilde{P}$ is random permutations, hence the only chance that the adversary can differ these two games is one of following events occurs:

- There are queries $(\mathsf{SK}_1; H_0)$ and $(\mathsf{SK}_1; H_0)$ such that $\mathsf{SK}_1 \neq \mathsf{SK}_2$ and $\tilde{H}_0(\mathsf{SK}_1) = \tilde{H}_0(\mathsf{SK}_2)$;
- There are queries $(\mathsf{PK}_1, \mathsf{M}_1; \mathsf{DE})$ and $(\mathsf{PK}_2, \mathsf{M}_2; \mathsf{DE})$ such that $(\mathsf{PK}_1, \mathsf{M}_1) \neq (\mathsf{PK}_2, \mathsf{M}_2)$ but they satisfy either $\mathsf{C}_1^1 = \mathsf{C}_1^2$ or $\mathsf{C}_2^1 = \mathsf{C}_2^2$ where $(\mathsf{C}_1^\ell, \mathsf{C}_2^\ell) = \widetilde{\mathsf{DE}}(\mathsf{PK}_\ell, \mathsf{M}_\ell)$;

– There are queries $(\mathsf{PK}_1, \mathsf{M}_1; \mathsf{DE})$ and $(\mathsf{PK}_2, \mathsf{M}_2; \mathsf{DE})$ such that $(\mathsf{PK}_1, \mathsf{M}_1) \neq (\mathsf{PK}_2, \mathsf{M}_2)$ but they satisfy either $\mathsf{C}_1^1 = \mathsf{C}_1^2$ or $\mathsf{C}_2^1 = \mathsf{C}_2^2$ where $(\mathsf{C}_1^\ell, \mathsf{C}_2^\ell) = E(\mathsf{PK}_\ell, \mathsf{M}_\ell)$;

We observe that if none of the events occurs then we can replace the honest interfaces with (F, E, D), which represents Game 8. Moreover, we can bound the probability of those bad events as:

$$\Pr[\mathsf{Bad}] \leq \frac{2q^2}{|\mathcal{X}|} + \frac{3q^2}{|\mathcal{Y}|}$$

which refers to

$$|\Pr[\text{Game 7}] - \Pr[\text{Game 8}]| \leq \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Combining all claims together, we complete the entire proof. □

G Proof of Theorem 8

In this section, we give the full proof of Theorem 8, that we show Π_{PKE} is indifferentiable from an ideal PKE.

Proof. According to the definition of indifferentiability, we immediately observe that, the adversary has three honest interfaces (PKG, PE and PD) and four adversarial interfaces ($H_0, \mathsf{DKG}, \mathsf{DE}$ and DD). Therefore, we need to build an efficient simulator $\mathcal{S}$ that can simulate the six adversarial interfaces properly, which means, 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, there exists a system that responds to all of the queries(both honest and adversarial) in a slightly different way and then we build our simulator $\mathcal{S}$ as the system in the last game. Before the description of the games, we first specify some parameters:

– there are seven types of queries such as $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$, $(\mathsf{sk}, \mathsf{DKG})$, $(\mathsf{pk}, \mathsf{M}, \mathsf{R}; \mathsf{DE})$, $(\mathsf{sk}, \mathsf{C}; \mathsf{DD})$, $(\mathsf{SK}, \mathsf{PKG})$, $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$, $(\mathsf{SK}, \mathsf{C}; \mathsf{PD})$, where $\mathsf{sk}, \mathsf{SK} \leftarrow \mathcal{X}$, $\mathsf{pk}, \mathsf{PK} \leftarrow \mathcal{Y}$, $\mathsf{R} \leftarrow \mathcal{R}$ and $\mathsf{C} \leftarrow \mathcal{C}$;
– adversary makes q queries to the system, where $q = \mathsf{poly}(\lambda)$;
– the real oracles used in the real world are $\tilde{H}_0, \widetilde{\mathsf{DKG}}, \widetilde{\mathsf{DE}}, \widetilde{\mathsf{DD}}, \widetilde{\mathsf{PKG}}, \widetilde{\mathsf{PE}}, \widetilde{\mathsf{PD}}$;
– in each game, the system's responses are denoted as $H_0^{\mathsf{r}}, \mathsf{DKG}^{\mathsf{r}}, \mathsf{DE}^{\mathsf{r}}, \mathsf{DD}^{\mathsf{r}}, \mathsf{PKG}^{\mathsf{r}}$, PE^{r} and PD^{r}, for instance, we denote $H_0^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$ as the system's response when adversary makes a query $Q = (\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$.

Next, we define the hybrid games. Specifically,

Game 0. This game is identical to the real game except that the system maintains three tables, referring to H_0-table, DE-table and DD-table. Concretely, the system responds to the queries the same as in the real world, for instance, $H_0^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}), \mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}) = \widetilde{\mathsf{PKG}}(\mathsf{SK})$ and so forth. For the tables, the system maintains them as follows:

- H_0-table: Initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{R}, \mathsf{r})$. Once the adversary makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$ which does not exist in H_0-table, the system inserts $(\mathsf{PK}, \mathsf{M}, \mathsf{R}, \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R})$ into H_0-table;
- DE-table: Initially empty, consists of tuples with form of $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{C})$. Once the adversary makes a query $(\mathsf{pk}, \mathsf{M}, \mathsf{R}; \mathsf{DE})$ which does not exist in DE-table, the system inserts $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \widetilde{\mathsf{DE}}(\mathsf{pk}, \mathsf{M}, \mathsf{R})$ into DE-table;
- DD-table: Initially empty, consists of tuples with form of $(\mathsf{sk}, \mathsf{C}, \mathsf{M}, \mathsf{r})$. Once the adversary makes a query $(\mathsf{sk}, \mathsf{C}; \mathsf{DD})$ which does not exist in DD-table, the system inserts $(\mathsf{sk}, \mathsf{C}, \widetilde{\mathsf{DD}}(\mathsf{sk}, \mathsf{C}))$ into DD-table.

Note that at this point all the queries are responded by the real oracles and these tables are just keeping track of information related to adversary'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.

Same as above, we here also define a relation between query Q and the table Tab. Concretely, if Q is a H_0 query ($Q = (\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$), we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ in Tab such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}, T_3 = \mathsf{R}$. Analogously, if Q is a DE query $Q = (\mathsf{pk}, \mathsf{M}, \mathsf{R}; \mathsf{DE})$, we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{Tab}$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_3 = \mathsf{R}$; if Q is a DD query, say $Q = (\mathsf{sk}, \mathsf{C}; \mathsf{DD})$, we say $Q \in \mathsf{Tab}$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4)$ such that $T_1 = \mathsf{sk}$ and $T_2 = \mathsf{C}$.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries. Specifically,

H_0-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$, then the system responds as follows:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$ and $T_3 = \mathsf{R}$, then the system responds with T_4;
- Case 2. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$-table such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}, T_3 = \mathsf{r}||0^{n_3}$ and $T_4 = \widetilde{\mathsf{PE}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$, then the system responds with r;
- Case 3. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{PK}, T_3 = \mathsf{M}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = T_2$ and $T_4 = \mathsf{r}||0^{n_3}$, then the system responds with r;
- Case 4. Otherwise, the system responds with $\tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R})$ and inserts the tuple $(\mathsf{PK}, \mathsf{M}, \mathsf{R}, \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}))$ into H_0-table.

DKG-QUERY. Suppose $Q_k = (\mathsf{sk}, \mathsf{DKG})$, then the system responds with $\mathsf{PKG}^{\mathsf{r}}(\mathsf{sk})$.
DE-QUERY. Suppose $Q_k = (\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{DE})$(for ease, we split $\mathsf{R} = \mathsf{R}_1||\mathsf{R}_2$), then the system responds:

- Case 1. If $Q_k \in \mathsf{DE}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4)$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_3 = \mathsf{R}$, then the system responds with T_4;
- Case 2. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 \neq 0^{n_3}$, then

1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{pk}$, $T_3 = M$ and $T_4 = R$, then the system responds with T_2;
2. else, the system responds to Q_k with $\widetilde{\mathsf{DE}}(\mathsf{pk}, \mathsf{M}, \mathsf{R})$.

– Case 3. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 = 0^{n_3}$, then
1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}(T_1) = \mathsf{pk}$, $T_3 = M$ and $T_4 = R$, then the system responds with T_2;
2. If there is $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_4 = \mathsf{R}_1$, then the system responds with $\mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, T_3)$;
3. Otherwise the system responds to Q_k with $\widetilde{\mathsf{DE}}(\mathsf{pk}, \mathsf{M}, \mathsf{R})$.

DD-QUERY. Suppose $Q_k = (\mathsf{sk}, \mathsf{C}; \mathsf{DD})$, then the system responds,

– Case 1. If $Q_k \in \mathsf{DD}$, which means there exists a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$ such that $T_1 = \mathsf{sk}$ and $T_2 = \mathsf{C}$, then the system responds with $T_3||T_4$;
– Case 2. If $Q_k \notin \mathsf{DD}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$ such that $T_1 = \mathsf{PKG}(\mathsf{sk}), T_4 = \mathsf{C}$, then the system responds with $(T_2||T_3)$;
– Case 3. Otherwise, the system makes a query $(\mathsf{sk}, \mathsf{C}; \mathsf{PD})$ and responds,
1. If $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) = \perp$, then the system responds with $\perp$;
2. else if there is a tuple $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{PKG}^{\mathsf{r}}(\mathsf{sk}), T_2 = \mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})$ and $\mathsf{PE}^{\mathsf{r}}(T_1, T_2, T_3) = \mathsf{C}$, then the system responds to Q_k with $= (\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})||T_4||0^{n_3})$;
3. else, the system responds with $\widetilde{\mathsf{DD}}(\mathsf{sk}, \mathsf{C})$.

Game 2. This game is identical to Game 1, except for responding to H_0 queries. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$, then the system responds,

– Case 1. If $Q_k \in H_0$, same as in Game 1;
– Case 2. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$-table such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}, T_3 = \mathsf{r}||0^{n_3}$ and $T_4 = \widetilde{\mathsf{PE}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$, then same as in Game 1;
– Case 3. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{PK}, T_3 = \mathsf{M}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = T_2$ and $T_4 = \mathsf{r}||0^{n_3}$, then same as in Game 1;
– Case 4. Otherwise, the system samples $\mathsf{r} \leftarrow \{0,1\}^{n_3}$, responds to Q_k with r and inserts $(\mathsf{PK}, \mathsf{MmR}, \mathsf{r})$ into H_0-table.

Game 3. This game is identical to Game 2, except for responding to DE queries. Suppose $Q_k = (\mathsf{pk}, \mathsf{M}, \mathsf{R}; \mathsf{DE})$(for ease, we split $\mathsf{R} = \mathsf{R}_1||\mathsf{R}_2$), then the system responds,

– Case 1. If $Q_k \in \mathsf{DE}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4)$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_3 = \mathsf{R}$, then same as in Game 2;
– Case 2. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 \neq 0^{n_3}$, then
1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{pk}$, $T_3 = M$ and $T_4 = R$, then same as in Game 2;

2. else, the system samples $\mathsf{C} \xleftarrow{\$} \mathcal{C}$, responds to Q_k with C, and inserts $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{C})$ into DE table.

– Case 3. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 = 0^{n_3}$, then
1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}(T_1) = \mathsf{pk}$, $T_3 = \mathsf{M}$ and $T_4 = \mathsf{R}$, then the system responds with T_2;
2. If there is $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_4 = \mathsf{R}_1$, then the system responds with $\mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, T_3)$;
3. Otherwise the system samples $\mathsf{R}^* \xleftarrow{\$} \{0,1\}^{2n_3}$, responds to Q_k with $\mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*)$, and inserts $(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*, \mathsf{R}_1)$ and $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*))$ into H_0 table and DE table, respectively.

Game 4. This game is identical to Game 3, except for responding to the DD queries. Suppose $Q_k = (\mathsf{sk}, \mathsf{C}; \mathsf{DD})$, then the system responds,

– Case 1. If $Q_k \in \mathsf{DD}$, which means there exists a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$ such that $T_1 = \mathsf{sk}$ and $T_2 = \mathsf{C}$, then the system responds with $T_3 || T_4$;
– Case 2. If $Q_k \notin \mathsf{DD}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$ such that $T_1 = \mathsf{PKG}(\mathsf{sk}), T_4 = \mathsf{C}$, then the system responds with $(T_2 || T_3)$;
– Case 3. Otherwise, the system makes a query $(\mathsf{sk}, \mathsf{C}; \mathsf{PD})$ and responds,
1. If $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) = \perp$, then the system responds with $\perp$;
2. else if there is a tuple $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{PKG}^{\mathsf{r}}(\mathsf{sk}), T_2 = \mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})$ and $\mathsf{PE}^{\mathsf{r}}(T_1, T_2, T_3) = \mathsf{C}$, then the system responds to Q_k with $= (\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) || T_4 || 0^{n_3})$;
3. else, the system samples $\mathsf{str} \xleftarrow{\$} \{0,1\}^{n_3}$, responds to Q_k with $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) || \mathsf{str} || 0^{n_3}$ and inserts $(\mathsf{sk}, \mathsf{C}, \mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}), \mathsf{str} || 0^{n_3})$ into DD table.

Game 5. In Game 4, the queries to the adversarial interfaces are answered by the tables which're maintained by the system and by making queries to the honest interfaces. The system never makes queries directly to $\tilde{H}_0, \widetilde{\mathsf{DKG}}, \widetilde{\mathsf{DE}}$ and $\widetilde{\mathsf{DD}}$; these oracles are *only* used to answer the $\mathsf{PKG}, \mathsf{PE}, \mathsf{PD}$ queries (either generated by the adversary or by the system's response to $H_0, \mathsf{DKG}, \mathsf{DE}, \mathsf{DD}$ queries). At this point, it is straightforward to show that we can replace $\mathsf{PKG}, \mathsf{PE}$ and PD with the ideal versions from Definition 6, resulting in Game 5.

We note that in Game 5, the system is efficient, and it responds to the adversarial interfaces just by keeping several tables and accessing the honest interfaces. Thus, we can build a simulator that responds to $H_0, \mathsf{DKG}, \mathsf{DE}, \mathsf{DD}$ queries exactly as the system does in Game 5. The result is that the view in Game 5 is identical to the ideal world and it suffices to prove that any adjacent games are computational indistinguishable.

Simulator In Ideal Game. Let $(\mathsf{KENGEN}, \mathsf{ENC}, \mathsf{DEC})$ be the function pair that samples from $\mathcal{T}_{\mathsf{PKE}}$, the simulator works as follows. Like the system in Game 8, the simulator also maintains three tables, referring to H_0-table, DE-table and DD-table. Concretely:

– H_0-table: initially empty, consists of tuples with form of $(\mathsf{PK}, \mathsf{M}, \mathsf{R}, \mathsf{r})$;

- DE-table: initially empty, consists of tuples with form of $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{C})$;
- DD-table: initially empty, consists of tuples with form of $(\mathsf{sk}, \mathsf{C}, \mathsf{M}, \mathsf{r})$.

By definition, the simulator $\mathcal{S}$ has access to the honest interfaces $(\mathsf{PKG}, \mathsf{PD}, \mathsf{PE})$, where $\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{PKEYGEN}(\mathsf{SK}), \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \mathsf{ENC}(\mathsf{PK}, \mathsf{M}, \mathsf{R}), \mathsf{DD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{C}) = \mathsf{DEC}(\mathsf{SK}, \mathsf{C})$. And for the adversarial queries, $\mathcal{S}$ works as the system in Game 5, by just using the tables and calling the honest interfaces.

H_0-QUERY. Suppose $Q_k = (\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$, then the system responds as follows:

- Case 1. If $Q_k \in H_0$, which means there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}$ and $T_3 = \mathsf{R}$, then the system responds with T_4;
- Case 2. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$-table such that $T_1 = \mathsf{PK}, T_2 = \mathsf{M}, T_3 = \mathsf{r}||0^{n_3}$ and $T_4 = \widetilde{\mathsf{PE}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$, then the system responds with r;
- Case 3. If $Q_k \notin H_0$, then the system makes a query $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{PE})$. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{PK}, T_3 = \mathsf{M}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = T_2$ and $T_4 = \mathsf{r}||0^{n_3}$, then the system responds with r;
- Case 4. Otherwise, the system samples $\mathsf{r} \xleftarrow{\$} \{0,1\}^{n_3}$, responds to Q_k with r and inserts $\mathsf{PK}, \mathsf{M}, \mathsf{R}, \mathsf{r}$ into H_0 table.

DKG-QUERY. Suppose $Q_k = (\mathsf{sk}, \mathsf{DKG})$, then the system responds with $\mathsf{PKG}^{\mathsf{r}}(\mathsf{sk})$.

DE-QUERY. Suppose $Q_k = (\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{DE})$(for ease, we split $\mathsf{R} = \mathsf{R}_1||\mathsf{R}_2$), then the system responds:

- Case 1. If $Q_k \in \mathsf{DE}$, which means there is a tuple $T = (T_1, T_2, T_3, T_4)$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_3 = \mathsf{R}$, then the system responds with T_4;
- Case 2. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 \neq 0^{n_3}$, then
  1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}^{\mathsf{r}}(T_1) = \mathsf{pk}$, $T_3 = M$ and $T_4 = R$, then the system responds with T_2;
  2. else, the system samples $\mathsf{C} \xleftarrow{\$} \mathcal{C}$, responds to Q_k with C and inserts $\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{C}$ into DE table.
- Case 3. If $Q_k \notin \mathsf{DE}$ and $\mathsf{R}_2 = 0^{n_3}$, then
  1. If there is a tuple $T = (T_1, T_2, T_3, T_4)$ in DD-table such that $\mathsf{PKG}(T_1) = \mathsf{pk}$, $T_3 = M$ and $T_4 = R$, then the system responds with T_2;
  2. If there is $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{pk}, T_2 = \mathsf{M}, T_4 = \mathsf{R}_1$, then the system responds with $\mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, T_3)$;
  3. Otherwise the system samples $\mathsf{R}^* \xleftarrow{\$} \{0,1\}^{2n_3}$, responds to Q_k with $\mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*)$, and inserts $(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*, \mathsf{R}_1)$ and $(\mathsf{pk}, \mathsf{M}, \mathsf{R}, \mathsf{PE}^{\mathsf{r}}(\mathsf{pk}, \mathsf{M}, \mathsf{R}^*))$ into H_0 table and DE table, respectively.

DD-QUERY. Suppose $Q_k = (\mathsf{sk}, \mathsf{C}; \mathsf{DD})$, then the system responds,

- Case 1. If $Q_k \in \mathsf{DD}$, which means there exists a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DD}$ such that $T_1 = \mathsf{sk}$ and $T_2 = \mathsf{C}$, then the system responds with $T_3||T_4$;
- Case 2. If $Q_k \notin \mathsf{DD}$ but there is a tuple $T = (T_1, T_2, T_3, T_4) \in \mathsf{DE}$ such that $T_1 = \mathsf{PKG}(\mathsf{sk}), T_4 = \mathsf{C}$, then the system responds with $(T_2||T_3)$;

– Case 3. Otherwise, the system makes a query $(\mathsf{sk}, \mathsf{C}; \mathsf{PD})$ and responds,
  1. If $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) = \perp$, then the system responds with $\perp$;
  2. else if there is a tuple $(T_1, T_2, T_3, T_4) \in H_0$ such that $T_1 = \mathsf{PKG}^{\mathsf{r}}(\mathsf{sk}), T_2 = \mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})$ and $\mathsf{PE}^{\mathsf{r}}(T_1, T_2, T_3) = \mathsf{C}$, then the system responds to Q_k with $= (\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})||T_4||0^{n_3})$;
  3. else, the system samples $\mathsf{str} \xleftarrow{\$} \{0,1\}^{n_3}$, responds to Q_k with $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C})||\mathsf{str}||0^{n_3}$ and inserts $(\mathsf{sk}, \mathsf{C}, \mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}), \mathsf{str}||0^{n_3})$ into DD table.

Now we are ready to prove the indistinguishability between any adjacent games.

Claim. Game Real $\approx$ Game 0.

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

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

Claim. Game 0 $\approx$ Game 1.

Proof. By definition, we note that in Game 1, the system maintains longer tables and it responds to part of the queries (type A query) by just using those tables and calling the honest interfaces. For the other queries (type B query), the system responds by calling the real oracles. However, there is one case (case 3.1 in DD) that might differ these two games. Specifically, we note that $\mathsf{PD}^{\mathsf{r}}(\mathsf{sk}, \mathsf{C}) = \perp$ does not implies $\widetilde{\mathsf{DD}}(\mathsf{sk}, \mathsf{C})$, in fact, there are two cases might cause abortion: 1) $\widetilde{\mathsf{DD}}(\mathsf{sk}, \mathsf{C})$ aborts; 2) $\widetilde{\mathsf{DD}}(\mathsf{sk}, \mathsf{C}) = \mathsf{M}, \mathsf{R}_1, \mathsf{R}_2$ where $\mathsf{R}_2 \neq 0^{n_3}$. Hence, it suffices to prove that, any adversary can not hand in a valid ciphertext $\mathsf{C} = \widetilde{\mathsf{DE}}(\mathsf{pk}, \mathsf{M}, \mathsf{R}_1||\mathsf{R}_2)(\mathsf{R}_2 \neq 0^{n_3})$ without making a DE query except with negligible probability. In fact, the honest interfaces would not help as

$$\mathsf{PD}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) := \widetilde{\mathsf{DD}}(\mathsf{PK}, \mathsf{M}, \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R})||0^{n_3})$$

where $R_2 \equiv 0^{n_3}$. Besides, $\widetilde{\mathsf{DE}}$ is a random injection, it's trivial that the probability that adversary can output a valid ciphertext is bounded by $\frac{q}{|\mathcal{Y}| \times |\mathcal{M}|}$, which refers to

$$|\Pr[\text{Game 0}] - \Pr[\text{Game 1}]| \leq \frac{q}{|\mathcal{Y}| \times |\mathcal{M}|}.$$

Claim. Game 1 $\approx$ Game 2.

Proof. Recalling that the only difference between Game 1 and Game 2 occurs in case 4. Suppose $Q_k = (\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*; H_0)$, in Game 1, the system responds to Q_k with $\tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$ while in Game 2 the system replaces the response with a random string str^*. To prove the indistinguishability, we first formalize the adversary's view in Game 1. By definition, we immediately observe that, in Game 1,

- The system responds to $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; H_0)$ with $\tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R})$;
- The system responds to $(\mathsf{PK}, \mathsf{M}, \mathsf{R}; \mathsf{DE})$ with $\tilde{\mathsf{DE}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$;
- The system responds to $(\mathsf{SK}, \mathsf{C}; \mathsf{DD})$ with $\tilde{\mathsf{DD}}(\mathsf{SK}, \mathsf{C})$.

Hence, in the adversary's perspective, under the consistency condition listed below, the responses of H_0 are independent and random strings. For DE query, the responses should be random injections and DD is the *inverse* of DE. Here are the consistency conditions:

- $\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$;
- $\mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}) = \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, H_0^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})||0^{n_3})$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})) = \mathsf{M}$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}||0^{n_3})) = \mathsf{M}$;
- $\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}_1||\mathsf{R}_2) = (\mathsf{M}, \mathsf{R}_1||\mathsf{R}_2)$;
- There exists no $(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) \neq (\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$ such that $\mathsf{DE}^r(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) = \mathsf{DE}^r(\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$.

Now we turn to Game 2, where the system responds to Q_k with str^* instead. Next, we explore the view on the adversarial interfaces. For H_0 query, in either game, the response is independent and random string, referring to the response is well distributed. For DE query, we note that after Q_k, the system implicitly sets $\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}) = \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$, hence, as long as the adversary does not make a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}; \mathsf{DE})$ before Q_k, the response is consistent. In fact, str^* is uniformly sampled, the probability of this bad event is bounded by $\frac{k-1}{2^{n_3}}$. For DD query, we have that after Q_k, the system implicitly sets $\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{DE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3})) = \mathsf{DD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)) = (\mathsf{M}, \mathsf{str}^*||0^{n_3})$. Due to definition, we note that case 3 implies that the adversary never make such a query before Q_k, hence the response is consistent in either game.

Next, we show that, with high probability, the consistency condition holds.

First equation. This equation holds trivially.

Second equation. Note that the only case that breaks this equation is that the adversary makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}; \mathsf{DE})$ before Q_k, which is bounded by $\frac{k-1}{2^{n_3}}$.

Third equation. This equation is independent of H_0 query, so it holds trivially in Game 2.

Fourth equation. Due to the second and third equation, we immediately observe that if for any $(\mathsf{PK}, \mathsf{M})$ and $\mathsf{str} \in \{0,1\}^{n_3}$, there exists $R \in \{0,1\}^{2n_3}$ such that $H_0^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R}) = \mathsf{str}$, then the fourth equation holds for certain. In fact, in either Game 1 or Game 2, the responses of H_0 queries are independent and random string, which implies the bad event (no pre-image for H_0^{r}) is bounded by $(1 - \frac{1}{2^{n_3}})^{2^{2n_3}} \leq (\frac{1}{e})^{2^{n_3}}$.

Fifth equation. Easy to note that after Q_k the system implicitly sets

$$\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)) = (\mathsf{M}^*, \mathsf{str}^*||0^{n_3})$$

hence as long as the adversary never makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}; \mathsf{DE})$ before Q_k, this equation is implied by the second one.

Sixth equation. By definition, we note that the only case that would induce a collision is that the adversary makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*); \mathsf{DE})$ or $(\mathsf{SK}^*; \widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)||0^{n_3}); \mathsf{DD})$. In fact, $\tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$ is independent of the adversary's view, additionally $\widetilde{\mathsf{DE}}$ is a random injection, it's trivial that the probability of these bad events is bounded by $\frac{q}{2^{n_3}}$.

Combing together, we can bound the union of the bad events by

$$\Pr[\mathsf{Bad}] \leq \frac{k-1}{2^{n_3}} + 2(\frac{1}{e})^{2^{n_3}} + \frac{q}{2^{n_3}}$$

which refers to

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

Claim. Game 2 $\approx$ Game 3.

Proof. Recalling that the only difference between Game 2 and Game 3 occurs in case 2.2 and 3.3. Suppose $Q_k = (\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}_1^*||\mathsf{R}_2^*; \mathsf{DE})$, in Game 1, the system responds with $\widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}_1^*||\mathsf{R}_2^*)$ while in Game 2 the system replaces the response with a random ciphertext $\mathsf{C}^* \xleftarrow{\$} \mathcal{C}$.

Analogous to the analysis above(Game 1$\approx$ Game 2), we know that under the condition that the bad events above, the adversary's view on Game 2 is: the responses of H_0 queries are independent and random strings; DE^{r} is a random injection and DD^{r} is its inverse. Moreover, the responses also satisfy the equations:

- $\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$;
- $\mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}) = \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, H_0^r(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})||0^{n_3})$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})) = \mathsf{M}$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}||0^{n_3})) = \mathsf{M}$;
- $\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}_1||\mathsf{R}_2) = (\mathsf{M}, \mathsf{R}_1||\mathsf{R}_2)$;
- There exists no $(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) \neq (\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$ such that $\mathsf{DE}^r(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) = \mathsf{DE}^r(\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$.

Next we turn to **case 2.2** in Game 3, where the system responds to Q_k with C^*, and see the view on the adversarial interfaces. For H_0 queries, it's trivial that the responses are identical in both games. For DE queries, due to C^* is uniformly sampled, the responses are well distributed. For DD query, note that after Q_k, the system implicitly sets $\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{C}) = (\mathsf{M}^*, \mathsf{R}_1^*||\mathsf{R}_2^*)$, hence as long as the adversary never makes such a query before Q_k, the response is consistent(trivially bounded by $\frac{q}{|\mathcal{C}|}$). Moreover in case 2.2, $\mathsf{R}_2^* \neq 0^{n_3}$, the first four equations holds for free. Next we prove that the fifth and sixth equation also hold.

Fifth equation. Trivial to note that, if the adversary never makes a query $(\mathsf{SK}^*, \mathsf{C}^*; \mathsf{DD})$ before Q_k, the fifth equation holds for certain.

Sixth equation. As C^* is uniformly sampled, the probability of collision is bounded by $\frac{q}{|\mathcal{C}|}$.

Now we turn to **case 3.3** in Game 3, where $\mathsf{R}_2^* = 0^{n_3}$. For H_0 queries, we note that after Q_k, the system sets $H_0^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \hat{\mathsf{R}}) = \mathsf{R}_1^*$ if $\mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \hat{\mathsf{R}}) = \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$. However, R_1^* is chosen by the adversary, which means the response $H_0^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \hat{\mathsf{R}})$ is not well distributed. Therefore, we have to bound the probability of such a query. In fact, R^* is randomly sampled and hidden from the adversary, and $\widetilde{\mathsf{DE}}$ is an injection, which means the probability that adversary can make such a query is bounded by $\frac{q}{2^{n_3}}$. For DE query, note that after Q_k the system implicitly sets

$$\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}_1^* || 0^{n_3}) = \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*) = \widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*) || 0^{n_3}),$$

Note that $\widetilde{\mathsf{DE}}$ is a random injection, and R^* is uniformly chosen by the system, thus $\tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$ is independent of adversary view except with negligible probability($\leq \frac{q}{2^{n_3}}$), which implies the response to Q_k is well-distributed. For DD query, note that after Q_k the system sets

$$\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)) = (\mathsf{M}^*, \mathsf{R}_1^* || 0^{n_3}),$$

hence under the condition that $\mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*))$ is independent of adversary's view, the responses of DD queries are consistent.

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

First equation. This equation holds trivially.

Second equation. The only case that breaks this equation is the adversary makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R})$ such that $H_0^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}) = \mathsf{R}_1^*$ or $\tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}) = \mathsf{R}_1^*$. Fortunately, $\tilde{H}_0$ is a random oracle, and in Game 3, H_0^{r} is either answered by a random string (H_0 table) or $\tilde{H}_0$(from the DD list), hence the probability that adversary can make such a query is bounded by $\frac{q}{2^{n_3}}$.

Third equation. This equation is independent of DE queries, thus it holds for free.

Fourth equation. This equation is implied by the third one[18].

Fifth equation. Easy to see that, under the condition that adversary never makes a query $(\mathsf{SK}^*, \mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*); \mathsf{DD})$ before Q_k, this equation holds for certain.

Sixth equation. Note that the only case that would induce a collision is that the adversary makes a query $(\mathsf{SK}^*; \widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*) || 0^{n_3}); \mathsf{DD})$ before Q_k which is bounded as above.

Combing together, we can bound the union of the bad events by

$$\Pr[\mathsf{Bad}] \leq \frac{2q}{|\mathcal{C}|} + \frac{2q}{2^{n_3}},$$

[18] For case 3.3, if the system replaces the response with a random ciphertext(as in case 2.2), then this equation falls apart, because the decryption PD^{r} would abort with high probability.

refering to

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

Claim. Game 3 $\approx$ Game 4.

Proof. Recalling that the only difference between Game 3 and Game 4 occurs in case 3.3. Suppose $Q_k = (\mathsf{SK}^*, \mathsf{C}^*)$, in Game 3 the system responds to Q_k with $\widetilde{\mathsf{DD}}(\mathsf{SK}^*, \mathsf{C}^*)$ while in Game 4 the system replaces the response with $(\mathsf{M}^*, \mathsf{str}^*||0^{n_3})$ where $\mathsf{M}^* = \mathsf{PD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{C}^*)$ and $\mathsf{str}^* \xleftarrow{\$} \{0,1\}^{n_3}$.

Analogous to the analysis above, we know that under the condition that the bad events above, the adversary's view on Game 2 is: the responses of H_0 queries are independent and random strings; DE^{r} is a random injection and DD^{r} is its inverse. Moreover, the responses also satisfy the equations:

- $\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}) = \mathsf{DKG}^{\mathsf{r}}(\mathsf{SK})$;
- $\mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}) = \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, H_0^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})||0^{n_3})$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{PE}^{\mathsf{r}}(\mathsf{PKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R})) = \mathsf{M}$;
- $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}||0^{n_3})) = \mathsf{M}$;
- $\mathsf{DD}^{\mathsf{r}}(\mathsf{SK}, \mathsf{DE}^{\mathsf{r}}(\mathsf{DKG}^{\mathsf{r}}(\mathsf{SK}), \mathsf{M}, \mathsf{R}_1||\mathsf{R}_2) = (\mathsf{M}, \mathsf{R}_1||\mathsf{R}_2)$;
- There exists no $(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) \neq (\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$ such that $\mathsf{DE}^{r}(\mathsf{PK}_1, \mathsf{M}_1, \mathsf{R}^1) = \mathsf{DE}^{r}(\mathsf{PK}_2, \mathsf{M}_2, \mathsf{R}^2)$.

Next, we turn to case 3.3 in Game 4. By definition, we have that $\mathsf{PD}^{\mathsf{r}}(\mathsf{SK}^*, \mathsf{C}^*) = \mathsf{M}^* \neq \perp$, hence there exists a random string r^* such that $\widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{r}^*||0^{n_3}) = \mathsf{C}^*$. As $\tilde{H}_0$ is a random oracle, there exists $\mathsf{R}^* \in \{0,1\}^{n_3}$ such that $\mathsf{PE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*) = \mathsf{C}^*$[19] except for negligible probability($\leq (\frac{1}{e})^{2^{n_3}}$).

Now we see the view on the adversarial interfaces. For H_0 query, after Q_k, the system sets $H_0^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \hat{\mathsf{R}}) = \mathsf{str}^*$ if $\mathsf{PE}^{\mathsf{r}}(\mathsf{PK}\cdot\mathsf{M}^*, \hat{\mathsf{R}}) = \mathsf{C}^*$; as str^* is uniformly sampled, the response is well-distributed. For DE query, after Q_k, the system sets $\mathsf{DE}^{\mathsf{r}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}) = \mathsf{C}^* = \widetilde{\mathsf{DE}}(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{r}^*||0^{n_3})$, hence as long as the adversary never makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}; \mathsf{DE})$ before $Q_k (\leq \frac{k-1}{2^{n_3}})$, the response is consistent and well-distributed. For DD query, the system just replaces $\tilde{H}_0(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{R}^*)$ with str^*, hence the response is well distributed.

For the equations, we observe that the first, third and fourth one hold for free. And if adversary never makes a query $(\mathsf{PK}^*, \mathsf{M}^*, \mathsf{str}^*||0^{n_3}; \mathsf{DE})$ before Q_k, the second and fifth ones also hold trivially. And for the last one, we see that the responses of DE queries is either $\mathsf{PE}^{\mathsf{r}}(\mathsf{PK}, \mathsf{M}, \mathsf{R})$ or a random string in $\mathcal{C}$, hence the probability of a collision is bounded by $\frac{q}{2^{n_3}} + \frac{q}{|\mathcal{C}|}$.

Combing together, we can bound the union of the bad events by

$$\Pr[\mathsf{Bad}] \leq \frac{k-1}{2^{n_3}} + \frac{q}{|\mathcal{C}|} + \frac{q}{2^{n_3}},$$

refering to

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

[19] The adversary might know such a nonce R^*

Claim. Game 4 $\approx$ Game 5.

Proof. Due to the definition, we note that in Game 4, the system responds all of the adversarial interfaces just using tables and honest interfaces, it never directly calls the real oracles. Besides, $(\widetilde{\mathsf{DKG}}, \widetilde{\mathsf{DE}}, \widetilde{\mathsf{DD}})$ is an ideal DPKE scheme and $\tilde{H}_0$ is a random oracle, the only chance that adversary can differ Game 4 and Game 5 is one of the following events occurs:

- Case 1. There are queries $(\mathsf{PK}, \mathsf{M}, \mathsf{R}_1; \mathsf{PE})$ and $((\mathsf{PK}, \mathsf{M}, \mathsf{R}_2; \mathsf{PE}))$ such that $\tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}_1) = \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}_2)$;
- Case 2. There is a $\mathsf{r} \in \{0,1\}^{n_3}$ and $(\mathsf{PK}, \mathsf{M})$ such that r has no pre-image, which means $\forall \mathsf{R} \in \{0,1\}^{2n_3}, \tilde{H}_0(\mathsf{PK}, \mathsf{M}, \mathsf{R}) \neq \mathsf{r}$.

And it's trivial to note that if none of the cases occurs then we can replace the honest interfaces $(\mathsf{PKG}, \mathsf{PE}, \mathsf{PD})$ with an ideal PKE. Moreover we can bound the probability of the bad cases as:

$$\Pr[\text{Case 1}] \leq \frac{q^2}{2^{n_3}};\ \Pr[\text{Case 2}] \leq (1 - \frac{1}{2^{n_3}})^{2^{2n_3}} \leq (\frac{1}{e})^{2^{n_3}},$$

which refers to

$$|\Pr[\text{Game 4}] - \Pr[\text{Game 5}]| \leq \mathsf{negl}(\lambda).$$

Combining together, we complete the whole proof. □

H Proof of Theorem 10

In this section, we give the full proof of Theorem 10, that we show Π_{Sig} is indifferentiable from an ideal signature scheme.

Proof. According to the definition of indifferentiability, we immediately observe that the adversary has three honest interfaces $(\mathsf{SKG}, \mathsf{SS}, \mathsf{SV})$ and six adversarial interfaces $(H_0, H_1, P, P^{-1}, E, E^{-1})$. Therefore, we need to build an efficient simulator $\mathcal{S}$ that can simulate the six adversarial interfaces H_0, H_1, P, P^{-1}, E and E^{-1} properly, which means, 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. Before the description of the games, we first specify some parameters and events:

- there are nine types of query: $(x, H_0), (m, H_1), (y, P), (y, P^{-1}), (y, m, z, E), (y, m, z, E^{-1}), (x, \mathsf{SKG}), (x, m, \mathsf{SS}), (y, m, \Sigma, \mathsf{SV}$ where $x \leftarrow \mathcal{X}, y \leftarrow \mathcal{Y}, m \leftarrow \mathcal{M}, z \leftarrow \mathcal{Z}$;
- adversary makes at most q queries to the system, where $q = \mathsf{poly}(\lambda)$;
- the real oracles used in the construction are $\tilde{H}_0, \tilde{H}_1, \tilde{P}, \tilde{P}^{-1}, \tilde{E}, \tilde{E^{-1}}$;
- the advantage of random-message attack is bounded by ϵ_1;
- the advantage of pseudorandom public key is bounded by ϵ_2.

Note that in the normal lazy sampling of random oracles or permutations, each output will be chosen essentially at random. However, a simulator for indifferentiability will need to occasionally sample in such a way as to be consistent with the ideal signature. Next we define two events, named "Strong Consistency Check"(SCC) and "Weak Consistency Check"(WCC). These checks look for the cases where the adversary may already have the necessary information to predict the query response without making the query. If the checks pass, it means the adversary is unable to make such a prediction, and the simulator is essentially free to answer randomly. On the other hand, if the checks fail, the simulator must answer carefully to be consistent with the adversary's prediction.

Strong Consistency Check. Let $Q_1, \ldots, Q_q$ be the sequence of the queries, we say event SCC occurs for the k-th query ($\mathsf{SCC}_k = 1$), if any one of the following cases is satisfied:

- Case 1. The k-th query is P query, say (y, P) and in the previous $k-1$ queries, there is no query with form of $(x, H_0), (x, \mathsf{SKG})$ or (x, m, SS) such that $\mathsf{SKG}(x) = y$.
  Note that if there had been a previous query on such an x, then it would be possible for the adversary to predict $P(y) = P(\mathsf{SKG}(x))$ without making a P query at all by just evaluating $\mathsf{keygen}(H_0(x))$.
- Case 2. The k-th query is P^{-1} query, say (y, P^{-1}) and in the previous $k-1$ queries, there exists no query with form of $(x, H_0), (x, \mathsf{SKG})$ or (x, m, SS) such that $\mathsf{keygen}(\tilde{H}_0(x)) = y$.
  Note that if there had been a previous query on such an x, the adversary can predict $P^{-1}(y) = P^{-1}(\mathsf{keygen}(\tilde{H}_0(x)))$ by querying $\mathsf{SKG}(x)$.

We note in the ideal world, the simulator is unable to tell if SCC happens, since doing so requires knowing the adversary's queries to $\mathsf{SKG}, \mathsf{SS}$ and SV. Instead, the simulator will be able to carry out a weak consistency check, WCC:

Weak Consistency Check. Let $Q_1, \ldots, Q_q$ be the sequence of the queries, we say event WCC occurs for the k-th query ($\mathsf{WCC}_k = 1$), if one of the following case satisfies:

- Case 1. The k-th query is P query, say (y, P) and in the previous $k-1$ queries, there is no query with form of (x, H_0) such that $\mathsf{SKG}(x) = y$.
- Case 2. The k-th query is P^{-1} query, say (y, P^{-1}) and in the previous $k-1$ queries, there is no query with form of (x, H_0) such that $\mathsf{keygen}(\tilde{H}_0(x)) = y$.
- Case 3. The k-th query is E query, say (y, m, z, E) and in the previous $k-1$ queries, there is no (m, H_1) query or no query with form of (x, H_0) such that $\mathsf{SKG}(x) = y$.
- Case 4. The k-th query is E^{-1} query, say (y, m, z, E^{-1}) and in the previous $k-1$ queries, there is no (m, H_1) query or no query with form of (x, H_0) such that $\mathsf{SKG}(x) = y$.

This weak consistency check will not catch all the bad cases, but we will demonstrate that the adversary is unable to generate such bad cases, except with negligible probability.

Now we are ready the define the games. After each game, we also give the intuition for why that game is indistinguishable from the previous game.

Game 0. This game is identical to the real game, where every query is answered by applying real oracles. For instance, the response of (x, H_0) is $\tilde{H}_0(x)$, the response of (x, SKG) is $\tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(x)))$ and so forth. Moreover, during the queries, it also maintains six tables, referring to H_0-table, H_1-table, P-table, P^{-1}-table, E-table and E^{-1}-table. Concretely,

- H_0-table: Initially empty, consists of tuples with form of (x, r, y, y'). Once the adversary makes a (x, H_0) query which does not exist in H_0-table (no tuple that the first element of it is x), it inserts $(x, \tilde{H}_0(x), \mathsf{keygen}(\tilde{H}_0(x)), \mathsf{SKG}(x))$ into the table.
- H_1-table: Initially empty, consists of tuples with form of (m, M). Once the adversary makes a (m, H_1) query that does not exist in H_1-table, it inserts $(m, \tilde{H}_1(m))$ into the table.
- P-table: Initially empty, consist of tuples with form of $(*, *, y, y')$. Once the adversary makes a (y, P) query which does not exist in P-table, it inserts $(*, *, \tilde{P}(y), y)$ into the table.
- P^{-1}-table: Initially empty, consist of tuples with form of $(*, *, y, y')$. Once the adversary makes a (y, P^{-1}) query which does not exist in P-table, it inserts $(*, *, y, \tilde{P}^{-1}(y))$ into the table.
- E-table: Initially empty, consist of tuples with form of (y, m, z, z'). Once the adversay makes a (y, m, z', E) query which does not exist in E-table, it inserts $(y, m, \tilde{E}_{y||m}(z'), z')$ into the table.
- E^{-1}-table: Initially empty, consist of tuples with form of (y, m, z, z'). Once the adversay makes a (y, m, z, E^{-1}) query which does not exist in E-table, it inserts $(y, m, z\tilde{E^{-1}}_{y||m}(z))$ into the table.

Note that at this point these tables are just keeping track of information relating to adversary's queries, and are completely hidden to the adversary. Next, same as above, we define a relation based on these tables. We will say a H_0 query $Q_k = (x, H_0)$ is in a table K, denoted $Q_k \in K$, if there is a 4-tuple in K such that the 1st element is equal to x. Analogously, for a P query we say $Q_k = (y, P) \in K$ if there is a 4-tuple in K such that the 4-th element is equal to y; for P^{-1} queries, we say $Q_k = (y, P^{-1}) \in K$ if there is a 4-tuple in K such that the 3rd element is equal to y. For a H_1-query, we say $Q_k = (m, H_1) \in K$ if there is a 2-tuple $T = (T_1, T_2)$in K such that $T_1 = m$; for an E-query, we say $Q_k = (y, m, z', E) \in K$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4) \in K$ such that $T_1 = y, T_2 = m, T_4 = z'$; for an E^{-1}-query, we say $Q_k = (y, m, z, E^{-1}) \in K$ if there is a 4-tuple $T = (T_1, T_2, T_3, T_4) \in K$ such that $T_1 = y, T_2 = m, T_3 = z$.

Then, we show a game that the system responds to the queries by using both tables and the real oracles without changing adversary's view.

Game 1. This game is identical to Game 0, except the way of maintaining the tables and responding to the queries. Specifically,

H_0-QUERY. Suppose (x, H_0) is the k-th query($k \in [1, q]$), the system responds as follows:

- Case 1. If $Q_k \in H_0$, then it responds with the corresponding output string;
- Case 2. If $Q_k \notin H_0$ but there is a tuple $T = (T_1, T_2, T_3, T_4)$ in P-table such that $T_4 = \mathsf{SKG}(x)$ and $T_2 \neq *$, then responds with T_2 and inserts (x, T_2, T_3, T_4) into H_0 table;
- Case 3. Otherwise, responds with $\tilde{H}_0(x)$ and inserts $(x, \tilde{H}_0(x), \mathsf{keygen}(\tilde{H}_0(x)), \mathsf{SKG}(x))$ and $(*, \tilde{H}_0(x), \mathsf{keygen}(\tilde{H}_0(x)), \mathsf{SKG}(x))$ into H_0-table and P-table, respectively.

P-QUERY. Suppose (y, P) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, then respond with the third element of the 4-tuple corresponding to Q_k;
- Case 2. Otherwise, responds with $\tilde{P}(y)$, and inserts $(*, *, \tilde{P}(y), y)$ into both P and P^{-1} table.

P^{-1}-QUERY. Suppose (y, P) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, then respond with the fourth element of the 4-tuple corresponding to Q_k;
- Case 2. Otherwise, responds with $\tilde{P}^{-1}(y)$, and inserts $(*, *, y, \tilde{P}^{-1}(y))$ into both P and P^{-1} table.

H_1-QUERY. Suppose (m, H_1) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_1$, then responds with the corresponding string;
- Case 2. Otherwise, the system responds with $\tilde{H}_1(m)$.

E-QUERY. Suppose (y, m, z', E) be the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then responds with 3rd element of the 4-tuple corresponding to Q_k;
- If $Q_k \notin E \cup E^{-1}$ and $\mathsf{SV}(y, m, z') = 0$, then the system responds with $\tilde{E}_{y||m}(z)$;
- If $Q_k \notin E \cup E^{-1}$ but $\mathsf{SV}(y, m, z') = 1$, then
  1. If there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P$ such that $T_4 = y, T_2 \neq *$ then the system responds with $\mathsf{sign}(T_2, H_1(m))(H_1(m)$ is extracted from H_1-table);
  2. Otherwise, it responds with $\tilde{E}_{y,m}(z')$

E^{-1}-QUERY. Suppose (y, m, z, E^{-1}) be the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then responds with 4th element of the 4-tuple corresponding to Q_k;

- If $Q_k \notin E \cup E^{-1}$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P^{-1}$, such that $T_4 = y, T_1 \neq *, T_1' = m$, then the system tests $\mathsf{ver}(T_3, H_1(m), z) \stackrel{?}{=} 1(H_1(m)$ is extracted from H_1-table). If so, it responds with $\mathsf{SS}(T_1, m)$, else $\tilde{E^{-1}}_{y,m}(z)$;
- Otherwise, the system responds $\tilde{E^{-1}}_{y,m}(z)$.

Note that, in Game 1 the system keeps a longer table, and for each query, it firstly checks whether it can be responded by the tables or honest interfaces. If can, then it responds with them, else calls the real oracles. In fact, the system in Game 1 is a prototype for our simulator $\mathcal{S}$, and in the following games, we replace the responses that answered by the real oracles with random strings step by step. Moreover, the tuples stored in the table are consistent of the real oracles, hence each response in Game 1 is identical to the one in Game 0.

Game 2. This game is identical to Game 1, except for answering P queries. Assuming (y, P) to be the k-th query, the system responds as follows:

- Case 1: Suppose $Q_k \in H_0 \cup P \cup P^{-1}$, same as above.
- Case 2: If there is no such tuple in $\notin H_0 \cup P \cup P^{-1}$, but $\mathsf{SCC}_k = 1$, then respond with a random string $y' \leftarrow \mathcal{Y}$. The system inserts $(*, *, y', y)$ into the P-table.
- Case 3: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$, same as above.

Note that strong consistency check determines whether or not the adversary has the ability to learn $\tilde{P}(y)$ by making a query to SKG. If so, we must answer the P query using $\tilde{P}$; otherwise we can answer randomly.

Game 3. This game is identical to Game 2, except modifying P-query once more. More concretely, in case 2, if $Q_k \notin P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then the system responds with a random public key $y' = \mathsf{keygen}(r)$, where $r \stackrel{\$}{\leftarrow} \mathcal{X}$ and inserts $(*, *, \mathsf{keygen}(r), y)$ into P-table.

Here, we just change the distribution of outputs in the case where we do not use $\tilde{P}$. Game 3 is indistinguishable from Game 2 by the pseudorandomness of the public key.

Game 4. This game is identical to Game 3, except the way of filling up the P-table. Concretely, in case 2, the system inserts the tuple $(*, r, \mathsf{keygen}(r), y)$ into the table.

Here, the only difference from Game 4 is that we record the random coins r used to sample the output of P into the P table. Since that r is not used with high probability , this is close to Game 4 from the adversary's perspective.

Game 5. This game is identical to Game 4, except that to answer P queries, where the system only uses the tables and honest interfaces. Assuming (y, P) to be the k-th query, then

- Case 1. $Q_k \in H_0 \cup P \cup P^{-1}$, same as above.
- Case 2. If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then same as above;

– Case 3. If If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0 \cap \mathsf{WCC}_k = 1$, then it samples $r \leftarrow \mathcal{X}$, responds with $\mathsf{keygen}(r)$ and inserts $(*, r, \mathsf{keygen}(r), y)$ into P-table.

Game 5 and Game 4 are identical unless the adversary makes a query Q_k such that $Q_k = (y, P), \mathsf{SCC}_k = 0, \mathsf{WCC}_k = 1$ and $Q_k \notin P \cup P^{-1}$. This means that $y = \mathsf{SKG}(x) = \tilde{P}^{-1}(\mathsf{keygen}(\tilde{H}_0(x)))$ for some x, and that the adversary previously queried (x, SKG) or (x, m, SS), but not (x, H_0). Since $Q_k \notin P \cup P^{-1}$, the adversary has never made a query on $(\mathsf{keygen}(\tilde{H}_0(x)), P^{-1})$, as such a query would have resulted in y being added to the table for P^{-1}. Therefore, $\tilde{H}_0(x)$ and thus $\mathsf{keygen}(\tilde{H}_0(x))$ are independent of the adversary's view. In Game 5, the query would result in $(*, r, \mathsf{keygen}(\mathsf{r}), y)$ being added to the table for P, meanwhile in Game 4, the table records $(*, *, \mathsf{keygen}(\tilde{H}_0(x)), y)$. However, with high probability r is only used after (x, H_0) query, therefore, after the corresponding H_0 query, the P-table also records $(*, r, \mathsf{keygen}(\mathsf{r}), y)$. The only difference is that in Game 5, $r = \tilde{H}_0$, whereas in Game 6, r is a fresh random value. However, since the adversary has no knowledge of $\tilde{H}_0(x)$, the two games are indistinguishable.

Game 6. This game is identical to Game 5, except the way it answers P^{-1} queries, where the system *only* uses the tables to simulate if the strong consistency check pass. Assuming (y, P^{-1}) to be the k-th query, then,

– Case 1: If $Q_k \in H_0 \cup P \cup P^{-1}$, same as above;
– Case 2: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then it samples $x \leftarrow \mathcal{X}$, and responds with $\mathsf{SKG}(x)$ and inserts $(x, *, y, \mathsf{SKG}(x))$ into P^{-1}-table.
– Case 3. Otherwise, the system responds with $\tilde{P^{-1}}(y)$.

Note that $\mathsf{SCC}_k = 0$ implies that $Q_k \in H_0$, hence the case $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0$ does not exist.

Due to the definition, Game 6 and Game 5 are identical unless there exists a query Q_k such that $Q_k = (y, P^{-1})$, $\mathsf{WCC}_k = 1$ and $Q_k \notin P \cup P^{-1}$. This means that, in the previous query, no x appears such that $y = \mathsf{keygen}(\tilde{H}_0(x))$, even in $\mathsf{SKG}, \mathsf{SS}$ queries, in other words, $\mathsf{SKG}(x)$ never appears and it is hidden from adversary's view. Thus we can replace it with a random value. Moreover, we can replace with $\mathsf{SKG}(\mathsf{x}')$ where $x' \xleftarrow{\$} \mathcal{X}$, as long $\mathsf{keygen}(\tilde{H}_0(x'))$ never appears in the queries. We note that, as x' is randomly sampled and hidden from the adversary, thus except with negligible probability, $\mathsf{keygen}(\tilde{H}_0(x'))$ never appears.

Game 7. This game is identical to Game 6, except the way it answers P^{-1} queries, where the system *only* uses the tables to simulate, and not the true oracle $\tilde{P}^{-1}$. Assuming (y, P^{-1}) to be the k-th query, then,

– Case 1: If $Q_k \in H_0 \cup P \cup P^{-1}$, same as above;
– Case 2: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 1$, then same as above.
– Case 3: If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{SCC}_k = 0, \mathsf{WCC}_k = 1$, then it samples $x \leftarrow \mathcal{X}$, and responds with $\mathsf{SKG}(x)$ and inserts $(x, *, y, \mathsf{SKG}(x))$ into P^{-1}-table.

Note that $\mathsf{WCC}_k = 0$ implies that $Q_k \in H_0$, hence the case $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{WCC}_k = 0$ does not exist.

Due to definition, Game 7 and Game 6 are identical unless there exists a query Q_k such that $Q_k = (y, P^{-1})$, $\mathsf{SCC}_k = 0, \mathsf{WCC}_k = 1$ and $Q_k \notin P \cup P^{-1}$. This means, there exist queries as (x, SKG) or $(x, m\mathsf{SS})$ such that $\mathsf{keygen}(\tilde{H}_0(x)) = y$. Moreover, by Game 5, we also note that query $(\mathsf{SKG}(x), P)$ would leak nothing about $\tilde{H}_0(x)$ and $\mathsf{keygen}(\tilde{H}_0(x))$. Hence, $\tilde{H}_0(x)$ and $\mathsf{keygen}(\tilde{H}_0(x))$ are hidden from the adversary's view. And if such an event happens, it means that the adversary is able to predict the public key for an unknown secret key. As our standard-model signature scheme has psedorandom public keys, this is impossible except with negligible probability.

Game 8. This game is identical to Game 7, except that to answer H_0 queries, where the system only uses the tables and honest interfaces. Assuming (x, H_0) is the k-th query, then

- Case 1. If $Q_k \in H_0$, same as above;
- Case 2. If $Q_k \notin H_0$ but there is a tuple $T = (T_1, T_2, T_3, T_4)$ in P-table such that $T_4 = \mathsf{SKG}(x)$ and $T_2 \neq *$, then same as above;
- Case 3. Otherwise, the system samples $r \leftarrow \mathcal{X}$, responds with r and inserts $(x, r, \mathsf{keygen}(r), \mathsf{SKG}(x))$ and $(*, r, \mathsf{keygen}(r), \mathsf{SKG}(x))$ into the H_0-table and P-table, respectively.

Recalling the H_0-table in Game 1, we immediately observe that, the only case that the system calls $\tilde{H}_0(x)$ is when the adversary knows nothing about $\tilde{H}_0(x)$ and $\mathsf{keygen}(\tilde{H}_0)(x)$, although the adversary might know $\mathsf{SKG}(x)$ and $(\mathsf{SKG}(x), P)$. Therefore, from the adversary's view, $\tilde{H}_0(x)$ is uniformly distributed in $\mathcal{X}$, and it is equivalent to randomly pick $r \leftarrow \mathcal{X}$ and implicitly set $r = (x, H_0)$ and $P^{-1}(\mathsf{keygen}(r)) = \mathsf{SKG}(x)$.

Game 9. This game is identical to Game 8, except that to answer H_1 query, where the system only uses the tables, not the true oracles. Assuming (m, H_1) is the k-th query, then

H_1-QUERY. Suppose (m, H_1) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_1$, then same as above;
- Case 2. Otherwise, the system randomly picks $M \leftarrow \mathcal{M}$, responds with M and inserts (m, M) into H_1-table.

Due to definition, we note that the only case that the system calls $\tilde{H}_1(m)$ is when the adversary knows nothing about $\tilde{H}_1(m)$, although the adversary might knows $\mathsf{SS}(x, m)$ for some secret key x. However, the adversary has not called $(\mathsf{SKG}(x), m, \mathsf{SS}(x, m), E)$ yet, otherwise (m, H_1) would be called automatically. Hence, $\tilde{H}_1(m)$ and $\mathsf{sign}(sk, \tilde{H}_1(m))$ are hidden from adversary's view and we can replace it with a random value.

Game 10. This game is identical to Game 9, expect that to answer E queries, where the system only uses the tables and honest interfaces. Assuming (y, m, z', E) is the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then responds with 3rd element of the 4-tuple corresponding to Q_k;
- If $Q_k \notin E \cup E^{-1}$ and $\mathsf{SV}(y, m, z') = 0$, then the system responds with $z \leftarrow \mathcal{Z}$,
- If $Q_k \notin E \cup E^{-1}$ but $\mathsf{SV}(y, m, z') = 1$, then the system responds with $\mathsf{sign}(T_2, H_1(m))$.

Game 11. This game is identical to Game 10, expect that to answer E^{-1} queries, where the system only uses the tables and honest interfaces. Assuming (y, m, z, E) is the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then same as above;
- If $Q_k \notin E \cup E^{-1}$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P^{-1}$, such that $T_4 = y, T_1 \neq *$, then the system tests $\mathsf{ver}(T_3, H_1(m), z) \stackrel{?}{=} 1$. If so, it responds with $\mathsf{SS}(T_1, m)$, else a random $z' \in \mathcal{Z}$;
- Otherwise, the system responds else a random $z' \in \mathcal{Z}$.

Game 12. In Game 11, the queries to H_0, H_1, P, P^{-1}, E and E^{-1} are answered by the tables which're maintained by the system and by making queries to $\mathsf{SKG}, \mathsf{SS}, \mathsf{SV}$. The system never makes queries directly to $\tilde{H}_0, \tilde{H}_1, \tilde{P}, \tilde{P^{-1}}, \tilde{E}, \tilde{E^{-1}}$; these oracles are *only* used to answer the $\mathsf{SKG}, \mathsf{SS}, \mathsf{SV}$ queries (either generated by the adversary or by the system's response to $H_0, H_1, P, P^{-1}, E, E^{-1}$ queries). At this point, it is straightforward to show that we can replace $\mathsf{SKG}, \mathsf{SS}$ and SV with the ideal versions from Definition 9, resulting in Game 12.

We note that in Game 12, the system is efficient, and it responds to the adversarial interfaces just by keeping several tables and accessing the honest interfaces. Thus, we can build a simulator that responds to $H_0, H_1, P, P^{-1}, E, E^{-1}$ queries exactly as the system does in Game 12. The result is that Game 12 corresponds to the ideal world. Since Game 0 is the real world, it suffices to prove that any adjacent games are indistinguishable. Next we will give the description of our simulator and prove the indistinguishability between each adjacent games.

Simulator. Let $\Pi_{\mathsf{Sig}} = (\mathsf{SKG}, \mathsf{SS}, \mathsf{SV})$ be an ideal signature scheme, associated with secret key space $\mathcal{X}$, public key space $\mathcal{Y}$, message space $\mathcal{M}$ and signature space $\mathcal{Z}$, where 1) $|\mathcal{X}| \leq |\mathcal{Y}|$; 2) $|\mathcal{M}| \leq |\mathcal{Z}|$. The simulator $\mathcal{S}$ responds to the queries as follows:

H_0-QUERY. Suppose (x, H_0) is the k-th query($k \in [1, q]$), the system responds as follows:

- Case 1. If $Q_k \in H_0$, then it responds with the corresponding output string;
- Case 2. If $Q_k \notin H_0$ but there is a tuple $T = (T_1, T_2, T_3, T_4)$ in P-table such that $T_4 = \mathsf{SKG}(x)$ and $T_2 \neq *$, then responds with T_2 and inserts (x, T_2, T_3, T_4) into H_0 table;
- Case 3. Otherwise, the system samples $r \leftarrow \mathcal{X}$, responds with r and inserts $(x, r, \mathsf{keygen}(r), \mathsf{SKG}(x))$ and $(*, r, \mathsf{keygen}(r), \mathsf{SKG}(x))$ into the H_0-table and P-table, respectively.

H_1-QUERY. Suppose (m, H_1) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_1$, then responds with the corresponding string;
- Case 2. Otherwise, the system randomly picks $M \leftarrow \mathcal{M}$, responds with M and inserts (m, M) into H_1-table.

P-QUERY. Suppose (y, P) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, then respond with the third element of the 4-tuple corresponding to Q_k;
- Case 2.If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{WCC}_k = 1$, then it samples $r \leftarrow \mathcal{X}$, responds with $\mathsf{keygen}(r)$ and inserts $(*, r, \mathsf{keygen}(r), y)$ into P-table.

P^{-1}-QUERY. Suppose (y, P) be the k-th query, the system responds as follows:

- Case 1. If $Q_k \in H_0 \cup P \cup P^{-1}$, then respond with the fourth element of the 4-tuple corresponding to Q_k;
- Case 2. If $Q_k \notin H_0 \cup P \cup P^{-1}$ and $\mathsf{WCC}_k = 1$, then it samples $x \leftarrow \mathcal{X}$, and responds with $\mathsf{SKG}(x)$ and inserts $(x, *, y, \mathsf{SKG}(x))$ into P^{-1}-table.

E-QUERY. Suppose (y, m, z', E) be the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then responds with 3rd element of the 4-tuple corresponding to Q_k;
- If $Q_k \notin E \cup E^{-1}$ and $\mathsf{SV}(y, m, z') = 0$, then the system responds with $z \leftarrow \mathcal{Z}$,
- If $Q_k \notin E \cup E^{-1}$ but $\mathsf{SV}(y, m, z') = 1$, then the system responds with $\mathsf{sign}(T_2, H_1(m))$.

E^{-1}-QUERY. Suppose (y, m, z, E^{-1}) be the k-th query, the system firstly calls queries (m, H_1) and (y, P), then it responds as follows:

- If $Q_k \in E \cup E^{-1}$, then responds with 4th element of the 4-tuple corresponding to Q_k;
- If $Q_k \notin E \cup E^{-1}$ and there is a tuple $T = (T_1, T_2, T_3, T_4) \in H_0 \cup P^{-1}$, such that $T_4 = y, T_1 \neq *$, then the system tests $\mathsf{ver}(T_3, H_1(m), z) \stackrel{?}{=} 1$. If so, it responds with $\mathsf{SS}(T_1, m)$, else a random $z' \in \mathcal{Z}$;
- Otherwise, the system responds else a random $z' \in \mathcal{Z}$.

Note that $\mathcal{S}$ works exactly the same as the system in Game 12. Moreover, the distribution of the honest interfaces in Game 12 is also identical to the ones in the ideal game, which means, for any adversary, the advantage in both Game 12 and Ideal Game is identical. Therefore, it suffices to prove that any adjacent games are indistinguishable.

Lemma 14. *Let Z be a variable such that $Z = \mathsf{sign}(r, m)$, where $r \leftarrow \mathcal{X}, m \leftarrow \mathcal{M}$, then for any $z \in \mathcal{Z}$, $\Pr[Z = z] \leq \sqrt{\epsilon_2}$.*

Proof. Assuming there exists z^* such that $\Pr[Z = z^*] > \sqrt{\epsilon_2}$, then it trivially breaks the RMA security by randomly samples m and r, and hands in its forgery as $\mathsf{sign}(r, m)$.

Claim. Game 0 $\approx$ Game 1.

Proof. It's trivial to note that Pr[Game 0] = Pr[Game 1], as in Game 1, the system only changes the way of maintaining the tables and the terms stored in each tuple is identical to the responses of real oracles.

Claim. Game 1 $\approx$ Game 2.

Proof. Firstly, we note that, in either Game 1 or Game 2, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 1 and Game 2 is when Case 3 occurs. Suppose (y, P) is the k-th query, the adversary's view on the adversarial interfaces are as follows:

View on H_0. By definition, $\tilde{H}_0$ is a random oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random.

Thus, the adversary's view on H_0 is consistent in both games, if the responses of H_0 queries satisfy the two properties above. When case 3 occurs, the responses of H_0 queries do not change as it always responds with $\tilde{H}_0$.

View on P. By definition, $\tilde{P}$ is a random permutation oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random; 3) the response is injective.

Thus, the adversary's view on P is consistent in both games, if the responses of P queries satisfy the three properties above. When case 3 occurs, after Q_k, the system sets $P(\mathsf{y}) = \mathsf{r}$. As r is randomly chosen, property 1, 2 hold trivially, and property 3 holds if there is no collision of P queries on r.

View on P^{-1}. By definition $\tilde{P^{-1}}$ is the inverse of a random permuatation oracle such that 1) the response is consistent; 2) the response is the inverse of P; 3) the response is injective; 4) $P^{-1}(\mathsf{keygen}(H_0(\mathsf{x}))) = \mathsf{SKG}(\mathsf{x})$.

Thus, the adversary's view on P^{-1} is consistent in both games, if the responses of P queries satisfy the four properties above. When case 3 occurs, after Q_k, the system sets $P^{-1}(r) = y$, which implies property 2. Moreover, if r never appears in the previous queries, and $\tilde{P}(\mathsf{y}), \tilde{P^{-1}}(\mathsf{r})$ never appear during the queries, then property 1 and 3 also hold. For property 4, And easy to note that property 4 holds if the adversary cannot output x^* such that $\mathsf{keygen}(\tilde{H}_0(x^*)) = \tilde{P}(y)$ or $\mathsf{keygen}(\tilde{H}_0(x^*)) = \mathsf{r}$. Applying Lemma 12, this bad event is bounded by $2q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}}$.

View on H_1. By definition, $\tilde{H}_1$ is a random oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random.

Thus, the adversary's view on H_1 is consistent in both games, if the responses of H_1 queries satisfy the two properties above. When case 3 occurs, the responses of H_1 queries do not change as it always responds with $\tilde{H}_1$.

View on E. Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E)$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 1$, then $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1$.
- 4) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then $E_{\mathsf{y},\mathsf{m}}(\mathsf{z})$ is uniformly random and $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 0$;

Thus, the adversary's view on E is consistent in both games, if the responses of E queries satisfy the four properties above. When case 3 occurs, we note that the responses of the queries are consistent, which means property 1 and 2 trivially hold. For property 4, as $\Pi_{\mathsf{SM-Sig}}$ is unique signature scheme; if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then

$$\Pr[\mathsf{ver}(\mathsf{r}, H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1] \leq \frac{1}{|\mathcal{Z}|}.$$

However, we note that, in game 2, $\tilde{P}(\mathsf{y})$ to r, then $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1$ fails immediately, hence we need bound the probability that $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 1$. In fact, as $\mathsf{SCC}_k = 1$, the adversary knows nothing of x such that $\mathsf{SKG}(\mathsf{x}) = y$, hence if $x, \tilde{H}_0(x)$ and $\mathsf{keygen}(\tilde{H}_0(x))$ never appears, then property 3 holds. Applying lemma 12, we can bound it by $\frac{q}{|\mathcal{X}|} + q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}}$.

View on E^{-1} Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E^{-1})$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 1$, then $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 1$.
- 4) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 0$, then $E^{-1}_{\mathsf{y},\mathsf{m}}(z)$ is uniformly random and $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 0$.

Thus, the adversary's view on E^{-1} is consistent in both games, if the responses of E^{-1} queries satisfy the four properties above. When case 3 occurs, we note that the responses of the queries are consistent, which means property 1 and 2 trivially hold. For property 4, same as above, we have that if $\mathsf{ver}(\tilde{P}(\mathsf{m}), H_1(\mathsf{m}), z) = 0$, then $\Pr[\mathsf{ver}(\mathsf{r}, H_1(\mathsf{m}), z) = 1] \leq \frac{1}{|\mathcal{Z}|}$. However, if $\mathsf{ver}(\tilde{P}(\mathsf{m}), H_1(\mathsf{m}), z) = 1$, then property 3 breaks immediately, hene we need bound the probability of the bad event. In fact, as we already assume $\tilde{H}_0$(the sign key of $\Pi_{\mathsf{SM-Sig}}$) never appears, this bad event can be bounded by $q\epsilon_2$, the RMA-security.

Now, we bound the union of these bad events as:

$$\Pr[\mathsf{Bad}] \leq \frac{2q}{|\mathcal{Y}|} + 3q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + \frac{2}{|\mathcal{Z}|} + \frac{q}{|\mathcal{X}|} + q\epsilon_2.$$

which referring to

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

Claim. Game 2 $\approx$ Game 3.

Proof. This proof is trivial as we just change the distribution of outputs from random strings to random public key. Hence, due to the pseudorandomness of keygen, we have

$$|\Pr[\text{Game 2}] - \Pr[\text{Game 3}]| \leq q\epsilon_1 \leq \mathsf{negl}(\lambda).$$

Claim. Game 3 $\approx$ Game 4.

Proof. In Game 4, the system only changes the way of maintaining the tables, and if r is not used, then the view in both games is consistent. Thus,

$$|\Pr[\text{Game 3}] = \Pr[\text{Game 4}]| \leq |\Pr[\text{Game 1}] = \Pr[\text{Game 2}]|.$$

Claim. Game 4 $\approx$ Game 5.

Proof. Firstly, we note that, in either Game 4 or Game 5, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 4 and Game 5 is when Case 3 occurs. Suppose (y, P) is the k-th query, where $\mathsf{y} = \mathsf{SKG}(\mathsf{x})$ and x is known by the adversary, then its view on the adversarial interfaces are as follows:

View on H_0. By definition, $\tilde{H_0}$ is a random oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random.

Thus, the adversary's view on H_0 is consistent in both games, if the responses of H_0 queries satisfy the two properties above. When case 3 occurs, the system implicitly sets $H_0(\mathsf{x}) = \mathsf{r}$, as r is randomly chosen, and (x, H_0) never appears in previous queries, property 1 and 2 holds trivially.

View on P. By definition, $\tilde{P}$ is a random permutation oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random; 3) the response is injective.

Thus, the adversary's view on P is consistent in both games, if the responses of P queries satisfy the three properties above. When case 3 occurs, after Q_k, the system sets $P(\mathsf{y}) = \mathsf{keygen}(r), \mathsf{r} \leftarrow \mathcal{X}$. Due to the pseudorandomness of the public key, property 1, 2 and 3 preserves.

View on P^{-1}. By definition $\tilde{P^{-1}}$ is the inverse of a random permuatation oracle such that 1) the response is consistent; 2) the response is the inverse of P; 3) the response is injective; 4) $P^{-1}(\mathsf{keygen}(H_0(\mathsf{x}))) = \mathsf{SKG}(\mathsf{x})$.

Thus, the adversary's view on P^{-1} is consistent in both games, if the responses of P queries satisfy the four properties above. When case 3 occurs, after

Q_k, the system sets $P^{-1}(\mathsf{keygen}(\mathsf{r})) = \mathsf{y}$, which implies property 2 and 4. If $\mathsf{keygen}(\mathsf{r})$ never appears in the previous queries, and $\tilde{P}(\mathsf{y})$, $\tilde{H}_0(\mathsf{x})$ and $\tilde{H}_0(\mathsf{x}^*)$ such that $\mathsf{keygen}(\mathsf{x}^*) = \mathsf{r}$ never appear, then property 1 and 3 also holds. By lemma 12, the probability of this event is bounded by $3q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{X}|}$.

View on H_1. Same as above.

View on E. Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E)$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 1$, then $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1$.
- 4) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then $E_{\mathsf{y},\mathsf{m}}(\mathsf{z})$ is uniformly random and $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 0$;

Thus, the adversary's view on E is consistent in both games, if the responses of E queries satisfy the four properties above. When case 3 occurs, the system sets $E_{\mathsf{y},\mathsf{m}}(\mathsf{z}) = \mathsf{sign}(\mathsf{r}, H_1(\mathsf{m}))$, which implies property 3. Moreover, if $\mathsf{sign}(\mathsf{r}, H_1(\mathsf{m}))$ never appears in the previous queries and $\mathsf{sign}(\tilde{H}_0\mathsf{x}, H_1(\mathsf{m}))\mathrm{m}$ then the response is consistent and injective. Applying Lemma 14, this bad event can be bounded by $2q\sqrt{\epsilon_2}$. For property 4, as $\Pi_{\mathsf{SM-Sig}}$ is unique signature scheme; if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then

$$\Pr[\mathsf{ver}(\mathsf{r}, H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1] \leq \frac{1}{|\mathcal{Z}|}.$$

View on E^{-1} Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E^{-1})$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 1$, then $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 1$.
- 4) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 0$, then $E^{-1}_{\mathsf{y},\mathsf{m}}(z)$ is uniformly random and $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 0$.

Thus, the adversary's view on E^{-1} is consistent in both games, if the responses of E^{-1} queries satisfy the four properties above. When case 3 occurs, the system sets $E^{-1}_{\mathsf{y},\mathsf{m}}(\mathsf{sign}(\mathsf{r}, H_1(\mathsf{m}))) = \mathsf{z}$,which implies property 3. For property 1, 2, 4, the analysis is same as above.

Now, we can bound the union of these bad events as:

$$\Pr[\mathsf{Bad}] \leq q\epsilon_1 + 3q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + \frac{q}{|\mathcal{X}|} + \frac{1}{|\mathcal{Z}|} + q\sqrt{\epsilon_2} \leq \mathsf{negl}(\lambda)$$

which referring to

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

Claim. Game 5 $\approx$ Game 6.

Proof. Firstly, we note that, in either Game 5 or Game 6, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 4 and Game 5 is when Case 2 occurs. Suppose (y, P^{-1}) is the k-th query and $\mathsf{SCC}_k = 1$, in Game 5 the system sets $P^{-1}(\mathsf{y}) = \mathsf{SKG}(\mathsf{x})$ where $\mathsf{x} \xleftarrow{\$} \mathcal{X}$, we note that adversary might know T such that $\mathsf{keygen}(\mathsf{T}) = \mathsf{y}$. Then the adversary's view on the adversarial interfaces are as follows:

View on H_0. In either Game 5 or Game 6, the responses of H_0 queries do not change.

View on P. By definition, $\tilde{P}$ is a random permutation oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random; 3) the response is injective.

Thus, the adversary's view on P is consistent in both games, if the responses of P queries satisfy the three properties above. We note that, in both games $\tilde{P}$ is hidden and P-queries are answered by other tables and the honest interfaces, hence except negligible collision, property 3 holds.

View on P^{-1}. By definition $\tilde{P^{-1}}$ is the inverse of a random permuatation oracle such that 1) the response is consistent; 2) the response is the inverse of P; 3) the response is injective; 4) $P^{-1}(\mathsf{keygen}(H_0(\mathsf{x}))) = \mathsf{SKG}(\mathsf{x})$.

Thus, the adversary's view on P^{-1} is consistent in both games, if the responses of P queries satisfy the four properties above. When case 2 occurs, after Q_k, the system sets

$$P^{-1}(\mathsf{y}) = \mathsf{SKG}(x) = \tilde{P^{-1}}(\mathsf{keygen}(\tilde{H}_0(\mathsf{x})))$$

If $\mathsf{keygen}(\tilde{H}_0(\mathsf{x}))$ never appears in the previous queries, then $\mathsf{SKG}(x)$ is uniformly fresh, which means property 1, 2, 4 hold. For property 3, we note that if no x^*, such that $\mathsf{SKG}(\mathsf{x}^*) = \mathsf{SKG}(\mathsf{x})$ or $\mathsf{keygen}(\tilde{H}_0(\mathsf{x}^*)) = \mathsf{T}$ appear, then the responses are injective. Applying Lemma 12, we can bound it by $3q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}}$.

View on H_1. Same as above.

View on E. Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E)$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 1$, then $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1$.
- 4) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then $E_{\mathsf{y},\mathsf{m}}(\mathsf{z})$ is uniformly random and $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 0$;

Thus, the adversary's view on E is consistent in both games, if the responses of E queries satisfy the four properties above. Note that, in both game, the responses of E queries are consistent, which means property 1 and 2 hold trivially.

Moreover, we note that if no x^* such that $\mathsf{keygen}(\tilde{H}_0(x^*)) = y$ and $\mathsf{sign}(\tilde{H}_0(x^*))$ appear then, property 3 and 4 also hold.

View on E^{-1}. Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E^{-1})$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 1$, then $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 1$.
- 4) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 0$, then $E^{-1}_{\mathsf{y},\mathsf{m}}(z)$ is uniformly random and $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 0$.

Thus, the adversary's view on E^{-1} is consistent in both games, if the responses of E^{-1} queries satisfy the four properties above. When case 2 occurs, the system implictely sets $E^{-1}_{\mathsf{y},\mathsf{m}}(\mathsf{sign}(T, H_1(\mathsf{m}))) = \mathsf{SS}(\mathsf{x}, \mathsf{m})$,which implies property 3 immediately. For property 1, 2, 4, the analysis is same as above.

Now, we can bound the union of these bad events as:

$$\Pr[\mathsf{Bad}] \leq \frac{q}{|\mathcal{Y}|} + 3q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + q\sqrt{\epsilon_2} \leq \mathsf{negl}(\lambda),$$

which referring to

$$|\Pr[\text{Game 5}] - \Pr[\text{Game 6}]| \leq q \cdot \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Claim. Game 6 $\approx$ Game 7.

Proof. Suppose $Q_k = (\mathsf{y}, P^{-1})$ is the k-th query and we denote event Bad as $\mathsf{SCC_k} = 0 \cap \mathsf{WCC_k} = 1$. Next, we bound the bad event.

Assuming Bad occurs, then there is a x known by the adversary such that $y = \mathsf{keygen}(\tilde{H_0}(\mathsf{x}))$ and (x, H_0) never appears in previous $k-1$ queries. As $\tilde{H}_0(\mathsf{x})$ is unifromly distributed in $\mathcal{X}$, the probability that it can output a proper y is bounded by the pseudorandomness of the public key. By definition, it's trivial to note that

$$|\Pr[\text{Game 6}] - \Pr[\text{Game 7}]| \leq q \Pr[\mathsf{Bad}] \leq q\epsilon_1 \leq \mathsf{negl}(\lambda).$$

Claim. Game 7 $\approx$ Game 8.

Proof. Firstly, we note that, in either Game 7 or Game 8, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, the only difference between Game 7 and Game 8 is when Case 3 occurs. Suppose (x, H_0) is the k-th query, in Game 8 the system sets $H_0(\mathsf{x}) = \mathsf{r}$ where $\mathsf{r} \xleftarrow{\$} \mathcal{X}$. While we note that adversary might know $\mathsf{SKG}(\mathsf{x})$ before Q_k, with the restriction that $\mathsf{SKG}(\mathsf{x})$ never appears in $P \cup P^{-1}$ in the previous queries. Then the adversary's view on the adversarial interfaces are as follows:

View on H_0. By definition, $\tilde{H}_0$ is a random oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random.

Thus, the adversary's view on H_0 is consistent in both games, if the responses of H_0 queries satisfy the two properties above. When case 3 occurs, the system implicitly sets $H_0(\mathsf{x}) = \mathsf{r}$, as r is randomly chosen, and (x, H_0) never appears in previous queries, property 1 and 2 holds trivially.

View on P and P^{-1}. In both games, P and P^{-1} are answered by H_0 table and the honest interfaces. As the view on H_0 does not change, except with negligible collision, the view on P and P^{-1} preserves.

View on H_1. Same as above.

View on E and E^{-1}. Similar to the analysis above, we have that, if $\mathsf{r}, \mathsf{keygen}(r)$ and $\mathsf{sign}(\mathsf{r}, *)$ never appears in the previous queries and $\tilde{H}_0(\mathsf{x}), \mathsf{keygen}(\tilde{H}_0(\mathsf{x}))$ and $\mathsf{sign}(\tilde{H}_0(\mathsf{x}), *)$ never appears then the view on E and E^{-1} is consistent in both games. Applying Lemma 12 and 14, we can bound this bad event by $\frac{2q}{|\mathcal{X}|} + 2q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + 2q\sqrt{\epsilon_2}$.

Now, we can bound the union of these bad events as:

$$\Pr[\mathsf{Bad}] \leq \frac{q}{|\mathcal{Y}|} + 2q\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} + \frac{2q}{|\mathcal{X}|} + 2q\sqrt{\epsilon_2} \leq \mathsf{negl}(\lambda)$$

which referring to

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

Claim. Game 8 $\approx$ Game 9.

Proof. Firstly, we note that, in either Game 8 or Game 9, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, we note that in Game 9, the system replaces the responses of (m, H_1) with a random string M.

View on H_0, P, P^{-1}. It's trivial that, in either Game 8 or Game 9, the H_0, P, P^{-1} queries are responded in the same way, hence the view does not change.

View on H_1. By definition, $\tilde{H}_1$ is a random oracle such that 1) the response is consistent; 2) the response to a fresh input is uniformly random.

Thus, the adversary's view on H_1 is consistent in both games, if the responses of H_1 queries satisfy the two properties above. As M is randomly sampled, if M never appears in the previous queries, then property 1 and 2 holds.

View on E and E^{-1}. Similar to the analysis above, we note that if $\mathsf{M}, \mathsf{sign}(*, \mathsf{M})$ never appear in the previous queries and $\tilde{H}_0(\mathsf{m}), \mathsf{sign}(*, \tilde{H}_0(\mathsf{m}))$ never appear, then view on E and E^{-1} is consistent. In fact, $\Pi_{\mathsf{SM-Sig}}$ has unique signature, we can easily bound the bad event by $\frac{4q}{|\mathcal{M}|}$. Thus, we have

$$|\Pr[\text{Game 8}] - \Pr[\text{Game 9}]| \leq \frac{q^2}{|\mathcal{M}|} \leq \mathsf{negl}(\lambda).$$

Claim. Game 9 $\approx$ Game 10.

Proof. Firstly, we note that, in either Game 9 or Game 10, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, we note that in Game 10, the system replaces the responses, which fails in the check, with a random string z'.

View on H_0, H_1, P, P^{-1}. It's trivial that, in either Game 9 or Game 10, the H_0, H_1, P, P^{-1} queries are responded in the same way, hence the view does not change.

View on E. Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E)$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 1$, then $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1$.
- 4) if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then $E_{\mathsf{y},\mathsf{m}}(\mathsf{z})$ is uniformly random and $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 0$;

Thus, the adversary's view on E is consistent in both games, if the responses of E queries satisfy the four properties above. Property 3 holds trivial and if z' never appears in the previous queries, property 1 holds. Moreover, if $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, then

$$\Pr[\mathsf{ver}(\mathsf{r}, H_1(\mathsf{m}), E_{\mathsf{y},\mathsf{m}}(\mathsf{z})) = 1] \leq \frac{1}{|\mathcal{Z}|}.$$

Property 2 holds trivially if no collision occurs.

View on E^{-1} Suppose $(\mathsf{y}, \mathsf{m}, \mathsf{z}, E^{-1})$ to be an E query, then by definition, the view on E would satisfy the following:

- 1) the response is consistent;
- 2) the response is injective;
- 3) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 1$, then $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 1$.
- 4) if $\mathsf{ver}(P(\mathsf{m}), H_1(\mathsf{m}), z) = 0$, then $E^{-1}_{\mathsf{y},\mathsf{m}}(z)$ is uniformly random and $\mathsf{SV}(\mathsf{y}, \mathsf{m}, E^{-1}_{\mathsf{y},\mathsf{m}}(z)) = 0$.

As in Game 10, the system only replaces the value when $\mathsf{SV}(\mathsf{y}, \mathsf{m}, \mathsf{z}) = 0$, hence if z' never appears in the previous queries and $\tilde{E}_{\mathsf{y},\mathsf{m}}(\mathsf{z})$ never appears, then the view on E^{-1} preserves.

Now, we can bound the union of these bad events as:

$$\Pr[\mathsf{Bad}] \leq \frac{2q+1}{|\mathcal{Z}|} + \frac{q}{|\mathcal{Z}|} \leq \mathsf{negl}(\lambda)$$

which referring to

$$|\Pr[\text{Game 9}] - \Pr[\text{Game 10}]| \leq q \cdot \Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda).$$

Claim. Game 10 $\approx$ Game 11.

Proof. Firstly, we note that, in either Game 10 or Game 11, the adversary's view on the honest interfaces is consistent, hence it suffices to prove that, with high probability, the view on the adversarial interfaces also does not change. According to the definition, we note that in Game 11, the system replaces the responses, which fails in the check, with a random string.

View on H_0, H_1, P, P^{-1}. It's trivial that, in either Game 8 or Game 9, the H_0, H_1, P, P^{-1} queries are responded in the same way, hence the view does not change.

Moreover, in both games, $\tilde{H_0}, \tilde{P}, \tilde{P^{-1}}$ has been completely hidden, hence the probability that $\mathcal{A}$ outputs $(\mathsf{y}, \mathsf{m}, \mathsf{z})$ such that z is a valid signature under $\tilde{H_0}, \tilde{H_1}\tilde{P}, \tilde{P^{-1}}$ is bounded by the uniqueness of the signature scheme. Therefore, it's rest to show that, if the query $(\mathsf{y}, \mathsf{m}, \mathsf{z})$ fails in the check, it would not change the view on property 3, 4 whp. In other words, suppose$(\mathsf{y}, \mathsf{m}, \mathsf{z}, E^{-1})$ is the k-th query, then z is not the valid signature under H_0, P, P^{-1}, whp.

It's trivial that if z fails the validity check would not change the view, else then the adversary is asked to output a valid signature without knowing both sign key. Thus, we have

$$|\Pr[\text{Game 10}] - \Pr[\text{Game 11}]| \leq \frac{q}{|\mathcal{M}|} + q\epsilon_2 \leq \mathsf{negl}(\lambda).$$

Claim. Game 11 $\approx$ Game 12.

Proof. Due to definition, we note that in Game 11, the system responds all of the adversarial interfaces just using tables and honest interfaces, it never directly calls the real oracles. Besides, $\tilde{H_0}, \tilde{H_1}$ are random oracles, $\tilde{P}, \tilde{P^{-1}}$ are random permutations and $\tilde{E}, \tilde{E^{-1}}$ are ideal cipher models, hence the only chance the adversary can differ Game 9 and Game 10 is it can outputs $\mathsf{x} \neq \mathsf{x}' \in \mathcal{X}$ such that $\mathsf{keygen}(\tilde{H_0}(\mathsf{x})) = \mathsf{keygen}(\tilde{H_0}(\mathsf{x}'))$, which is bounded by the pseudorandomness of public key. Hence

$$|\Pr[\text{Game 11}] - \Pr[\text{Game 12}]| \leq q^2\sqrt{2\epsilon_1 + \frac{1}{|\mathcal{X}|}} \leq \mathsf{negl}(\lambda).$$

Combining together, we complete the whole proof. □