Indifferentiable Signatures: High Performance and Fallback Security

Charalampos Papamanthou
University of Maryland
cpap@umd.edu

Cong Zhang
University of Maryland
czhang20@umd.edu

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

May 16, 2021

Abstract

Digital signatures have been widely used as building blocks for constructing complex cryptosystems. To facilitate the security analysis of a complex system, we expect the underlying building blocks to achieve desirable composability. Notably, Canetti (FOCS 2001) and then Maurer et al (TCC 2004) propose analysis frameworks, the Universal Composability framework for cryptographic protocols, and the indifferentiability framework for cryptographic objects.

In this paper, we develop a "lifting strategy", which allows us to compile multiple existing practical signature schemes using cyclic group (e.g., Schnorr, Boneh-Boyen), to achieve a very stringent security guarantee, in an idealized model of the generic (bilinear) group, without introducing much extra efficiency loss. What's more interesting is that, in our design, even the involved idealized model does not exist, our compiled construction will still be able to achieve the classical notion of unforgeability.

To achieve both indifferentiability and good efficiency, we develop new techniques in generic (bilinear) group model.

1 Introduction

Provable security and composability. In modern cryptography, we are taking a rigorous approach: (new) constructions of cryptographic schemes and protocols should come with *proof of security*; and such proofs show that a cryptographic scheme satisfies a given definition of security, under certain assumptions. Different approaches towards defining security have been developed and evolved in the past decades, with game-based (or property-based) security definitions and composable security definitions being the two most prevalent ones. Composable security definitions are desirable since they offer modular design and analysis.

Several such frameworks have been developed, such as the Universal Composability (UC) framework by Canetti [Can01] for analyzing cryptographic (interactive) protocols; see also [PW01]. Important variants have also been introduced, and we list here some of them [CDPW07, MR11, MR16]. While these frameworks have been developed with different focuses in mind they all follow the *real-world/ideal-world simulation paradigm* [GM84, GMR89] for defining security. Inspired by Canetti's UC framework, Maurer, Renner, and Holenstein (MRH) [MRH04] then introduced the indifferentiability framework, which can support the modular way of design/analysis for cryptosystems and cryptographic objects.

Design and analysis in indifferentiability framework. The indifferentiability framework of a random system provides an alternative way to study the security. In this framework, a public primitive f is available, and the goal is to build another primitive F from f via a construction Π^f in order to securely replace its ideal counterpart F in variant environments. Since then, many fundamental cryptographic primitives have been designed and analyzed in the indifferentiability framework. Among these works, Coron et al. [CDMP05] were the first to suggest this approach to design hash functions that "behave like" random oracles in a provable sense. Specifically, indifferentiability assures that such a construction for hash function will provide any security that a random oracle (RO) achieves. In addition to random oracle model, more primitives such as ideal cipher, authenticated encryption have been investigated [DP06, DP07, CHK$^+$16, DKT16, DS16, BF18]. Those efforts have been extended to public key primitives such as public-key encryption (PKE) very recently by Zhandry and Zhang [ZZ20].

Design and analysis for digital signatures. The design and analysis of digital signatures is the focus of this paper. Digital signatures [DH76] are among the earliest public key primitives and are widely used today, from public key infrastructure (PKI) and authenticated communication and storage, to consensus and blockchains. Digital signatures can also be used as a building block for constructing more complex cryptographic primitives such as group signatures, ring signatures, anonymous credentials, authenticated non-interactive key-exchange, and many more.

In practice, many elegant designs exist. Notably, Schnorr [Sch91] presents a very efficient scheme which can be proven secure (i.e., unforgeable), in the random oracle model. Digital signature schemeshave also been standardized [NIS94] by NIST decades ago.

Very recently, digital signatures have been investigated in the indifferentiability framework, initiated by Zhandry and Zhang [ZZ20], and improved by Zhang and Zhou [ZZ21b]. While the indifferentiable signatures have been proposed are built in the random oracle model [ZZ21b] via the tree-based strategy, and the constructions are *not practical* (e.g. the signature size is over 1M bytes as shown in Table 1). In this paper, we expect to see better solutions.

Our design/analysis principles, and main question. In this paper, we aim to design "better" digital signatures. We have the following principles:

- First, the signature must be *easy to use.* That is, we should be able to compose the signature with other arbitrary primitives, without sacrificing security.

- Second, the signature design must be *robust.* If our signature achieves indifferentiability due to the use of an ideal primitive and this ideal primitive does not exist, the security of our signature should gracefully deteriorate to the classical notion of unforgeability.

- Finally, our design must be *truly efficient*, only introducing minimal extra cost.

At this point, we ask the main research question:

Is that possible to develop digital signatures which stratify the above three principles?

Ideally, we plan to develop a "lifting strategy", starting with existing practical signature schemes, and compiling these schemes to achieve indiffentiability in idealized models, without introducing much extra efficiency loss; again, even when the involved idealized models do not exist, the compiled version will still be able to achieve the classical notion of unforegability that the given practical signature schemes already achieved.

1.1 Our results

We give affirmative answers to the question above and the following is the overview of our results.

Practical and indifferentiable signature from GGM. We provide a construction for indifferentiable signature in the generic group model (GGM). More concretely, we carefully modify the Schnorr signature by replacing the cryptographic group with generic group model and show that this natural variant of the Schnorr signature can be proven indifferentiable from the ideal signature.

Practical and indifferentiable signatures from GBM. We give three constructions for indifferentiable signature in the generic bilinear-group model (GBM). Specifically, we start with Boneh-Boyen signatures, and carefully modify the signatures by replacing the cryptographic group with generic bilinear-group model, and prove that all the schemes are indifferentiable from ideal signature. Moreover, two of the constructions yield to optimal signature size.

Fallback security. We propose a new *fallback security* guarantee for digital signature schemes. Roughly speaking, when a scheme is provably secure in an idealized model, say generic group model for achieving a very stringent security notion, the fallback security guarantee ensures that, even when the generic group model is replaced with cryptographic groups in the standard model, the basic security notion (i.e., unforgeability) of the scheme still holds. For instance, we show that our modified version of Boneh-Boyen signature scheme enjoys indifferentiability in the generic bilinear-group model and at the same time the modified scheme remains unforgeable in standard model groups. In the paper, several signature schemes we present can ensure fallback security guarantee.

We here emphasize that our constructions are efficient; the schemes do not suffer from any signature size loss and it only increases one or two hash operations for the computation (signing or verification). Therefore we claim that our constructions satisfy the "three principles" for signature design. To help the readers to have a better sense, in Table 1 we illustrate a comparison among the schemes and show that our constructions achieve indifferentiability and fallback security along with high performance.

	Performance			Security		
	Size	Signing cost	Verification cost	Indifferentiability	Fallback	Fallback (ROM)
Schnorr Sig [Sch91] (EC)	64 Bytes	1 EXP+1 Hash	2EXP + 1Hash	No	-	-
BB-Sig [BB04]	64 Bytes	1 EXP	1 Pairing + 2 EXP	No	-	-
weak BB-Sig [BB04]	32 Bytes	1 EXP	1 Pairing + 1EXP	No	-	-
tree-based[1] Sig [ZZ21b]	1M Bytes	32K Hash	96K Hash	Yes	Yes	Yes
i-Schnorr Sec. 3	64 Bytes	1 EXP+3 Hash	1 EXP+2 Hash	Yes	No	Yes
i-BB Sec. 4.2	64 Bytes	1 EXP + 2 Hash	1 Pairing + 2 EXP + 1 Hash	Yes	Yes	Yes
i-weak BB_1 Sec. 4.4	32 Bytes	1EXP + 1 Hash	1 Pairing + 1 EXP + 1 Hash	Yes	No	No
i-weak BB_2 Sec. 4.4	32 Bytes	1 EXP + 2 Hash	1 Pairing + 1 EXP + 2 Hash	Yes	No	Yes

Table 1: This table illustrates a comparison among the digital signature schemes on both performance and security. We consider 128-bit security as suggested by "Cryptographic Key Length Recommendation"[2]. Here "EXP" denotes one exponentiation operation in groups, e.g. finite field; "Pairing" denotes one pairing operation in bilinear groups e.g. elliptic curve ; "Hash" denotes one hash operation, e.g. SHA3 or one block cipher operation, e.g. AES256. In addition "Fallback" and "Fallback (ROM)" means fallback security in standard model and in random oracle model, respectively.

[1]For the tree-based signature in [ZZ21b], we set the parameters as follows: 1) the length of the hash output is 256-bit; 2) there are 256 secret keys for Lamport's signature; 3) the height of the Merkle tree is 256.

[2]See https://www.keylength.com/en/3/

1.2 Technique Overview

To achieve the results listed above, we have to resolve multiple technical difficulties. In this part, we give an overlook on our techniques. As the techniques in building indifferentiable Boneh-Boyen signatures are similar with the one in indifferentiable Schnorr signature, we here only go over the techniques in building indifferentiable Schnorr signature.

Our starting point is the Schnorr signature scheme $\mathsf{Sch} = (\mathsf{Sch}.\textsc{Gen}, \mathsf{Sch}.\textsc{Sign}, \mathsf{Sch}.\textsc{Verify})$, where the signing key is a random value $sk \in \mathbb{Z}_p$ and the verification key is set to be $pk \leftarrow g^{sk}$ (g is a generator for a group). When signing a message m, the algorithm samples r, computes g^r, $e = \mathbf{H}(m, g^r)$ ($\mathbf{H}$ is a random oracle model) and $s = r - sk \cdot e$, and outputs signature (s, e).

One straightforward way to build an indifferentiable signature is to apply the techniques in [ZZ20], by combining random oracle model and Schnorr signature. However, Zhang and Zhou [ZZ21b] discover an attack, called dishonest public key attack, which breaks the indifferentiability of construction in [ZZ20]. They then give a solution to eliminate the dishonest public keys in the random oracle model, by carefully setting the space of the secret key much larger than the space of public key. Associated with this setting, Zhang and Zhou build an indifferentiable signature scheme in random oracle model. However, their solution relies on Lamport's signature and follows the tree-based construction, which means that their scheme is very impractical (shown in Table 1).

To build efficient constructions, we achieve this through a fundamentally different approach than that in [ZZ21b]; instead of using random oracle model, we consider more powerful idealized models, such as generic group model (GGM) and generic bilinear-group model (GBM). We clarify that, in our construction, we also use random oracle models and ideal cipher models. We argue that it is fine because in the framework of indifferentiability, Zhandry and Zhang [ZZ21a] prove that generic group model is strictly stronger than random oracle model and Holenstein et. al. [HKT11] show that random oracle model are equivalent to ideal cipher model (or random permutation model). Let $\mathbf{GGM} = (\mathbf{Label}, \mathbf{Add})$ be a generic group model and before we turn to our construction for indifferentiable Schnorr signature, we first recall the high level intuition of indifferentiability.

1.2.1 What is indifferentiability?

The seminal work of Maurer, Renner, and Holenstein [MRH04] proposes a sufficient and necessary security notion for composition of a random system. Concretely, in this framework, a public primitive f is available and the goal is to build another primitive F from f via a construction Π, say $F = \Pi^f$, and the security of indifferentiability requires that: there exists a simulator $\mathcal{S}$, such that the two systems (Π^f, f) (the real-world) and $(F, \mathcal{S}^F)$ (the ideal-world) are indistinguishable, even when the differentiator has access to f.

1.2.2 Making Gen indifferentiable

Following the ideas from [ZZ20], a proper way to construct a key generation algorithm is that $\textsc{Gen}(SK) = PK \leftarrow \mathbf{Label}(\mathbf{H}(SK))$, where $\mathbf{H}$ is a random oracle model. Besides, to prevent the dishonest public key attack shown in [ZZ21b], we also set the space of secret key to be much larger than the space of public key space.

Unfortunately, this construction fails in the generic group model. According to the definition of generic group model [Sho97], $\mathbf{Label}$ is a random injection maps $\mathbb{Z}_p$ to S, where the size of the codomain S is much larger than p. In other words, if randomly sampling a value $s^* \twoheadleftarrow S$, then with high probability, there exists no $x \in \mathbb{Z}_p$ such that $\mathbf{Label}(x) = s^*$ (we say s has no pre-image). And in such a case, we note that s^* is a dishonest public key. Even worse, no matter how large the space of the secret key is, dishonest public key exists, and we then show another attack based on this dishonest public key.

Due to definition, we know that if s^* has no pre-image, then $\mathbf{Add}(s^*, \cdot)$ would always output $\perp$. The following is the attack for the differentiator $\mathcal{D}$:

- Step 1: $\mathcal{D}$ samples SK, and makes two queries $PK \leftarrow \textsc{Gen}(SK)$; $s_1 \leftarrow \mathbf{Label}(1)$;
- Step 2: $\mathcal{D}$ samples s^* from S;
- Step 3: $\mathcal{D}$ flips a coin b; if $b = 0$ then $\mathcal{D}$ makes a query $T \leftarrow \mathbf{Add}(s_1, s^*)$ and otherwise makes a query $T \leftarrow \mathbf{Add}(s_1, PK)$. Set $b' = 0$ if and only if $T = \perp$;
- Step 4: outputs 1 if $b = b'$.

Trivial to note that, in the real-world, if s^* has no pre-image (with overwhelming probability), then $\mathcal{D}$ outputs 1. However, in the ideal-world, in the simulator's view, s^* and PK are statistically close, thus $\mathcal{D}$ outputs 0 within a noticeable probability ($\approx \frac{1}{2}$).

Observe that, to prevent this attack, we have to develop new techniques to eliminate those dishonest public keys. Roughly speaking, in our construction, we use a restricted generic group model (defined in Section 2), where we stress that $|S| = p$. As a result, it's trivial that, with high probability, every $s^* \in S$ has pre-image. Although this setting prevents the attack above, the simulation of $\mathbf{Label}$ and $\mathbf{Add}$ is subtle, and details can be found in Section 3.

1.2.3 Making Sign indifferentiable

Following the strategy from [ZZ20], we have a proper design for the signing algorithm, which takes (PK, SK, M, R) as inputs: $\textsc{Sign}(PK, SK, M, R) = V \leftarrow \mathbf{E}(PK||M, s||e)$, where $\mathbf{E}$ is an ideal cipher model, $PK||M$ is $\mathbf{E}$'s inverse key, and (s, e) is the Schnorr signature shown above. However, Zhang and Zhou [ZZ21b] propose an attack, called nonce-abuse attack, to break the indifferentiability if the randomness is not treated carefully. In the following, we show our design for treating randomness and show why it works.

A natural way to design the nonce is to set $r \leftarrow \mathbf{H}(R)$, where r is the seed used in Schnorr signature, and R is input nonce. However, this is insecure, and the following is an attack:

- Step 1: $\mathcal{D}$ samples SK, M, R and makes two queries $PK \leftarrow \textsc{Gen}(PK)$, $r \leftarrow \mathbf{H}(R)$;
- Step 2: $\mathcal{D}$ makes a query $V \leftarrow \textsc{Sign}(PK, SK, M, R)$;
- Step 3: $\mathcal{D}$ makes a query $(s, e) \leftarrow \mathbf{E}^{-1}(PK||M, V)$ and computes $sk \leftarrow \frac{r-s}{e}$;
- Step 4: $\mathcal{D}$ makes a query $T \leftarrow \mathbf{Label}(sk)$;
- Step 5: outputs 1 if and only if $PK = T$.

Trivial to note that, in real-world, $\mathcal{D}$ always outputs 1. However, in the ideal-world, the simulator knows nothing of SK and thus it cannot respond with a proper T and $\mathcal{D}$ outputs 0 with overwhelming probability.

Observe that, if $\mathcal{S}$ knows SK, then it can respond with a proper $T \leftarrow \textsc{Gen}(SK)$. With this in mind, we redesign the nonce to be $r \leftarrow \mathbf{H}(PK, SK, M, R)$. The reason why it works is that, once $\mathcal{D}$ wishes to have r, our strategy enforces $\mathcal{D}$ to leak SK to the simulator, and therefore the simulator can respond to the queries afterwards properly. In the following, we show how to build indifferentiable $\textsc{Sign}(PK, SK, M, R)$ (details can be found in Section 3):

$$\begin{aligned}
&r \leftarrow \mathbf{H}(PK, SK, M, R), e \leftarrow \mathbf{H}(M, \mathbf{Label}(r)), s \leftarrow r - sk \cdot e;\\
&V \leftarrow \mathbf{E}(PK||M, s||e).
\end{aligned}$$

Observe that, once settling down Gen and Sign, the algorithm Verify is fixed. Thus we complete our construction.

1.3 Organization

In the rest of our paper, Section 2 gives notations and definitions of the indifferentiability framework and additional definitions can be found in Appendix A. In Section 3 we build an indifferentiable Schnorr signature from generic group model, and present the proof; additional and detailed proof can be found in Appendix B. We then in Section 4 build two additional signatures, indifferentiable Boneh-Boyen signature in Section 4.2 and indifferentiable weak Boneh-Boyen signature in Section 4.4. In Section 5, we give an overview about related work.

2 Preliminaries

Notations. In the paper, $\lambda \in \mathbb{N}$ denotes the security parameter. For any integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{0, 1, \ldots, n-1\}$. For a non-empty finite set $\mathcal{X}$, we denote a *uniformly random* sample x from $\mathcal{X}$ as $x \twoheadleftarrow \mathcal{X}$. We overload this notation and extend it to probabilistic algorithms; in the paper, $y \twoheadleftarrow \mathcal{A}(x)$ means that y is assigned a value according to the distribution induced by algorithm $\mathcal{A}$ whose input value is x, and output value is y. When the algorithm $\mathcal{A}$ is deterministic, we write it as $y \leftarrow \mathcal{A}(x)$.

We use $x \overset{\text{re}}{\twoheadleftarrow} \mathcal{X}$ to denote that an element x is sampled from the set $\mathcal{X}$ by using the "rejection-resampling" strategy. More concretely, we let $\mathcal{Q}$ to denote the set of elements that have been previously sampled from the set $\mathcal{X}$; initially, $\mathcal{Q} := \emptyset$. To sample a *distinct and fresh* element x from $\mathcal{X}$, we first sample $x' \twoheadleftarrow \mathcal{X}$; if $x' \notin \mathcal{Q}$, then update $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x'\}$ and return $x \leftarrow x'$; otherwise, repeat the previous step.

When X and Y are strings, we write $X||Y$ to mean the string created by appending Y to X. We write $\{0,1\}^n$ for the set of all n-bit strings, where $n \in \mathbb{N}$.

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

2.1 Indifferentiability framework

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

2.1.1 Game playing technique.

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

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

Functionalities and games. Collections of procedures will sometimes implement particular abstract functionalities, for example that of some idealized primitive (e.g. a random oracle). A functionality is a collection $\mathcal{F} = (\mathcal{F}.hon, \mathcal{F}.adv)$; the names of these interfaces, hon and adv are suggestive as we will see in a moment. When games and adversaries are given access to a functionality a model of computation is defined. For example when the functionality is that of a random oracle, we have the random-oracle model. Thus one can think of functionalities and models somewhat interchangeably. As an example, functionality $\mathbf{RO} = (\mathbf{RO}.hon, \mathbf{RO}.adv)$, shown in Figure 1, implements a random oracle (with hon and adv interfaces) and will give rise to the random-oracle model.

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

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

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

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

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

By $\mathcal{G}^{\mathcal{F},\mathcal{A}}$ we denote a game using functionality $\mathcal{F}$ and adversary $\mathcal{A}$. If $\mathcal{F}'$ exports the same interface as $\mathcal{F}$, and adversary $\mathcal{A}'$ exports the same interface as $\mathcal{A}$, then $\mathcal{G}^{\mathcal{F}',\mathcal{A}'}$ executes the same game $\mathcal{G}$ with functional procedure $\mathcal{F}'$ and adversary $\mathcal{A}'$. We denote by $\mathcal{G}^{\mathcal{F},\mathcal{A}} \Rightarrow y$ the event that game $\mathcal{G}$ produces output y, that is procedure main returns value y. If game $\mathcal{G}$ uses any probabilistic procedure then $\mathcal{G}^{\mathcal{F},\mathcal{A}}$ is a random variable and by $\Pr[\mathcal{G}^{\mathcal{F},\mathcal{A}} \Rightarrow y]$ we denote the probability (over the combined randomness space of the game) that it takes on value y. Sometimes we need to make the random coins r explicit and write $\mathcal{G}^{\mathcal{F},\mathcal{A}}(r)$ to denote that the game is run on random coins r. Games are random variables over the entire random coins of the game and the adversarial procedures. For functionalities $\mathcal{F}$ and $\mathcal{F}'$ and adversaries $\mathcal{A}$ and $\mathcal{A}'$, we can thus consider the distance between the two random variables. Our security approach is that of concrete security, i.e., we say two games are ϵ-close if for all values y it holds that

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

2.1.2 Indifferentiability.

Fix two functionalities $\mathcal{F}_1$ and $\mathcal{F}_2$. A distinguisher $\mathcal{D}$ is an adversary that outputs a bit. A simulator is a procedure, usually denoted $\mathcal{S}$. Figure 2 defines two games $\mathbf{Real}$ and $\mathbf{Ideal}$. Fix some value y (e.g., $y = 1$).

The indifferentiability advantage of $\mathcal{D}$ is defined as

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

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

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

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

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

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

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

Figure 2: The games that define indifferentiability. Adversary $\mathcal{D}$ and functionalities $\mathcal{F}_1$, $\mathcal{F}_2$ are unspecified. The simulator $\mathcal{S}$ is a parameter of the game.

2.2 The idealized model for generic group

Generic group model (GGM) [Sho97]. For our purposes, a cryptographic group is a set $\mathcal{G}$ of prime size p, endowed with an efficiently computable group operation. Equivalently, a cryptographic group is a (not necesarily efficient) embedding of the additive group $\mathbb{Z}_p$ into some set. The Generic Group Model is an idealized model which assumes the existence of a random embedding from $\mathbb{Z}_p$. Concretely, a generic group is a pair $\mathbf{GGM} = (\mathbf{Label}, \mathbf{Add})$ where $\mathbf{Label}$ is a "labeling" function that is a random injection from $\mathbb{Z}_p$ to S, giving the embedding of $\mathbb{Z}_p$ into S, and $\mathbf{Add}$ is the induced group operation: $\mathbf{Add}(\mathbf{Label}(z_1), \mathbf{Label}(z_2)) = \mathbf{Label}(z_1 + z_2)$.

Now we describe functionality $\mathbf{GGM} = (\mathbf{GGM}.hon, \mathbf{GGM}.adv)$, as in Figure 3, which implements a generic group and will give rise to the generic-group model. Note that, two "adversarial" interfaces, $\mathbf{Label}.adv$ and $\mathbf{Add}.adv$ for capturing the permutation and the inverse, respectively, are defined so that the adversaries can access the functionality; and two "honest" interfaces, $\mathbf{Label}.hon$ and $\mathbf{Add}.hon$ are defined for all other procedures to access the functionality.

```
procedure GGM.$hon$:
INTERFACE Label.$hon(x)$:
If $x \notin \mathbb{Z}_p$
    then return $\perp$;
    else if $\mathbb{T}[x] = \perp$ then (re)sample $\mathbb{T}[x] \overset{\text{re}}{\twoheadleftarrow} S$;
return $\mathbb{T}[x]$.

INTERFACE Add.$hon(s_1, s_2)$:
If $\exists x_1, x_2 \in \mathbb{Z}_p$ so that
        Label.$hon(x_1) = s_1$ and Label.$hon(x_2) = s_2$
    then $x_3 \leftarrow x_1 + x_2 \bmod p$;
        return Label.$hon(x_3)$;
    else return $\perp$.

procedure GGM.$adv$:
INTERFACE Label.$adv(x)$:
return Label.$hon(x)$.

INTERFACE Add.$adv(s_1, s_2)$:
return Add.$hon(s_1, s_2)$.
```

Figure 3: Procedures implementing the functionality of the generic group model (GGM). The functionality is associated with $\mathbb{Z}_p$ and set S.

Remark 2.1. *In our paper, we will use a slightly strengthened version of the generic group model. Concretely, let S_{range}[3] be a public known subset of S such that $|S_{\mathsf{range}}| = p$, and the elements in S_{range} can be recognized efficiently, which means that, given an element $x \in S$, one can efficiently tell whether $x \in S_{\mathsf{range}}$ or not.*

In the strengthened GGM, now $\mathbf{Label}$ *is a "labeling" function that is a random injection from $\mathbb{Z}_p$ to S_{range}. The advantage of our new model is that, comparing to Shoup's model, in our new model, given an $x \in S$, we can easily tell whether x is a valid labeling or not.*

2.3 The idealized model for digital signatures

In Figure 4, we recall the functionality $\mathbf{SIG} = (\mathbf{SIG}.hon, \mathbf{SIG}.adv)$, which implements an ideal signature, that previously defined in [ZZ21b].

```
procedure SIG.hon

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

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

INTERFACE Verify.hon(PK, M, V):
If $\exists SK, R$ so that $\mathbf{Gen}.hon[SK] = PK$
            and $\mathbf{Sign}.hon[PK, SK, M, R] = V$
    then $\mathbb{T}[PK, M, V] \leftarrow 1$;
    else $\mathbb{T}[PK, M, V] \leftarrow 0$;
return $\mathbb{T}[PK, M, V]$.
```

```
procedure SIG.adv

INTERFACE Gen.adv(SK):
return $\mathbf{Gen}.hon(SK)$.

INTERFACE Sign.adv(PK, SK, M, R):
return $\mathbf{Sign}.hon(PK, SK, M, R)$.

INTERFACE Verify.adv(PK, M, V):
return $\mathbf{Verify}.hon(PK, M, V)$.
```

Figure 4: Procedures implementing the functionality of the ideal signature model. The associated parameters are verification key space $\mathcal{PK}$, signing key space $\mathcal{SK}$, message space $\mathcal{M}$, randomness space $\mathcal{R}$, and signature space $\mathbf{\Sigma}$.

In the ideal signature in Figure 4, three "honest" interfaces, $\mathbf{Gen}.hon$, $\mathbf{Sign}.hon$, $\mathbf{Verify}.hon$, are defined, for capturing key generation, signing and verification, respectively. Here, "adversarial" interfaces are identical to the honest ones. Several tables $\mathbb{T}[]$ have been used to trace the behaviors of the ideal signature. Notation "$\mathbb{T}[x] \leftarrow y$" means that, the value of the x-the record in the table is y; equivalently, for the query value x, the (potential) response value is y. In the ideal signature in Figure 4, the response values for PK and for signature V are randomly sampled. Whenever a signature is generated, the involved signing key SK must be well-defined, and be aware to the ideal signature.

[3]In the definition of Mareur's GGM [Mau05], S_{range} can be viewed as the handle space.

3 Indifferentiable Schnorr Signature

We here show how to compile and "lift" the original Schnorr signature scheme to achieve indifferentiability.

3.1 Schnorr signature

To do that, we first recall the Schnorr signature scheme $\mathsf{Sch} = (\mathsf{Sch}.\textsc{Gen}, \mathsf{Sch}.\textsc{Sign}, \mathsf{Sch}.\textsc{Verify})$, in the random oracle model; here let $\mathbf{H}$ denote the random oracle model as defined in Figure 1.

- Key generation $(PK, SK) \twoheadleftarrow \mathsf{Sch}.\textsc{Gen}(1^\lambda)$:
  $(\mathbb{G}, p, g) \twoheadleftarrow \mathsf{GroupGen}(1^\lambda)$; $x \twoheadleftarrow \mathbb{Z}_p$; $X \leftarrow g^x$; $SK \leftarrow x$; $PK \leftarrow X$; $\mathsf{params} \leftarrow (\mathbb{G}, p, g, \mathbf{H})$;
- Signing $V \twoheadleftarrow \mathsf{Sch}.\textsc{Sign}(\mathsf{params}, SK, M)$:
  $r \twoheadleftarrow \mathbb{Z}_p$; $R \leftarrow g^r$; $e \leftarrow \mathbf{H}(M, R)$; $s \leftarrow r - SK \cdot e$; $V \leftarrow (s, e)$;
- Verification $\phi \leftarrow \mathsf{Sch}.\textsc{Verify}(\mathsf{params}, PK, M, V)$:
  parse V into (V_1, V_2); $g^* \leftarrow g^{V_1} \cdot PK^{V_2}$; output $\phi = 1$ iff $V_2 = \mathbf{H}(M, g^*)$.

Correctness of Schnorr signature holds straightforwardly, essentially, $g^* = g^{V_1} \cdot PK^{V_2} = g^{r - SK\cdot e + SK \cdot e} = g^r$, which refers to $V_2 = \mathbf{H}(M, g^*)$. The security has been proven in [Sch91]: Schnorr signature scheme can achieve unforgeability in the random oracle, under the Discrete Logarithm assumption.

3.2 Indifferentiable Schnorr signature

We now show how to "lift" the original Schnorr signature scheme to achieve a much stronger security goal, ideal signature $\mathbf{SIG}$ as defined in Figure 4, at the price of relying on generic group model.[4]

Our construction is associated with public space $\mathcal{PK} = S$, where $|S| = p$, secret key space[5] $\mathcal{SK} = \{0,1\}^{\lceil \log p \rceil + \lambda}$, message space $\mathcal{M}$, nonce space $\mathcal{R} = \{0,1\}^{\lceil \log p \rceil + \lambda}$ and signature space $\mathbf{\Sigma} = \{0,1\}^t$. In our construction, we will use the following building blocks:

- $\mathbf{H}_{\text{sk}} : \{0,1\}^* \to [p]$ is a random oracle model.
- $\mathbf{H}_{\text{seed}} : \{0,1\}^* \to [p]$ is a random oracle model.
- $\mathbf{H}_{\text{test}} : \{0,1\}^* \to [p]$ is a random oracle model.
- $\mathbf{GGM} = (\mathbf{Label}, \mathbf{Add})$ is a generic group model that maps $\mathbb{Z}_p$ to S_{range}, where $|S_{\mathsf{range}}| = p$; see Figure 3.
- $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$ is a ideal cipher model, where $\mathbf{E} : \{0,1\}^k \times \{0,1\}^t \to \{0,1\}^t$, where the key length $k = \log|\mathcal{PK}| + \log|\mathcal{M}|$, and $t = 2\lceil \log p \rceil$, and $\mathbf{E}^{-1}$ is its inverse.

For ease of exposition, in the generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), we introduce additional notation; we denote $\mathbf{Scale}(\hat{g}, a) := \mathbf{Add}(\underbrace{\hat{g}, \cdots, \hat{g}}_{a})$, where $\hat{g}$ is a group element and $a \in \mathbb{Z}_p$.

[4]Note that, the generic group model implies the random oracle mode and the ideal cipher model.

[5]For the security of indifferentiability, we explicitly stress that the space of the secret key is much larger than the space of the public key.

```
$i\mathsf{Sch} = (i\mathsf{Sch}.\textbf{Gen}, i\mathsf{Sch}.\textbf{Sign}, i\mathsf{Sch}.\textbf{Verify})$

$PK \leftarrow i\mathsf{Sch}.\textsc{Gen}(SK)$:
On input, signing key $SK$, compute the verification key $PK$, as follows:
    $sk \leftarrow \mathbf{H}_{\text{sk}}(SK)$; $PK \leftarrow \mathbf{Label}(sk)$; return $PK$.

$V \leftarrow i\mathsf{Sch}.\textsc{Sign}(PK, SK, M, R)$:
On input, verification key $PK$, signing key $SK$, message $M$, nonce $R$, compute the signature $V$, as follows:
    If $PK = \textsc{Gen}(SK)$
        then $seed \leftarrow \mathbf{H}_{\text{seed}}(PK, SK, M, R)$;
            $g_{seed} \leftarrow \mathbf{Label}(seed)$;
            $e \leftarrow \mathbf{H}_{\text{test}}(PK, M, g_{seed})$;
            $s \leftarrow seed - \mathbf{H}_{\text{sk}}(SK) \cdot e$;
            output the signature $V \leftarrow \mathbf{E}\Big(PK||M,\ s||e\Big)$.
        else output the signature $V \leftarrow \perp$  //aborts if $PK$ is invalid

$\phi \leftarrow i\mathsf{Sch}.\textsc{Verify}(PK, M, V)$:
On input, verification key $PK$, message $M$ and signature $V$, operate as follows:
    1. "unpack" the signature $\Big(V_1, V_2\Big) \leftarrow \mathbf{E}^{-1}(PK||M,\ V)$;
    2. compute the corresponding group element:
       $g_{test} \leftarrow \mathbf{Add}\Big(\mathbf{Scale}(g, V_1), \mathbf{Scale}(PK, V_2)\Big)$;
    3. output $\phi \leftarrow 1$ if $V_2 = \mathbf{H}_{\text{test}}(PK, M, g_{test})$;
       output $\phi \leftarrow 0$, otherwise.
```

Correctness of $i\mathsf{Sch}$ follows easily:

$$\begin{aligned} g_{test} &= \mathbf{Add}\Big(\mathbf{Scale}(g, V_1), \mathbf{Scale}(PK, V_2)\Big) \\ &= \mathbf{Label}(seed - \mathbf{H}_{\text{sk}}(SK) \cdot V_2 + \mathbf{H}_{\text{sk}}(SK) \cdot V_2) = \mathbf{Label}(seed), \end{aligned}$$

which refers to $V_2 = \mathbf{H}_{\text{test}}(PK, M, g_{test})$.

In real-world, we can instantiate the generic group model $\mathbf{GGM} = (\mathbf{Label}, \mathbf{Add})$ by applying the elliptic curves in [JMV01], where g is a group generator. We set $\mathbf{Label}(x) := g^x$ and $\mathbf{Add}(g^x, g^y) := g^{x+y}$.

Remark 3.1. *To be consistent with the definition of ideal signature (Figure 4), our indifferentiable construction follows the query/response paradigm. Hence,* Gen*'s input is SK, rather than 1^λ;* Sign*'s inputs are (PK, SK, M, R), rather than (SK, M).*

Remark 3.2 (Fallback security)**.** *Our construction $i\mathsf{Sch}$ here enjoys a nice feature:*

- *when generic group model $\mathbf{GGM}$ does exist, our construction can achieve very strong security guarantee, i.e., indifferentiable from the ideal signatures;*
- *when generic group model $\mathbf{GGM}$ does not exist and boils down to ordinary cyclic group, our construction here can still be able to achieve the same security guarantee as that has been achieved by the original Schnorr signature; that is, our construction can achieve conventional unforgeability in the random oracle, assuming the discrete logarithm problem is hard over the cyclic group and $(\mathbf{E}, \mathbf{E}^{-1})$ is a keyed permutation only requiring correctness.*

3.3 Security

In this section, we prove that our construction achieves indifferentiable security in the generic group model, random oracle model, and ideal cipher model. Formally,

Theorem 3.3. *iSch = (iSch.GEN, iSch.SIGN, iSch.VERIFY) is indifferentiable from the ideal signature* $\mathbf{SIG} = (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$*, in the model for, random oracles* $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$*, generic group* $\mathbf{GGM} = (\mathbf{Label}, \mathbf{Add})$*, and ideal cipher* $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$*. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathsf{Adv}^{\mathsf{indiff}}_{i\mathsf{Sch},\mathbf{SIG},\mathcal{S}}(\mathcal{D}) \leq \frac{20q^2}{p} + 2p \cdot e^{-2^{\lambda}}.$$

Here, e is the natural logarithm and the simulator makes at most q^2 number of queries to its oracles.

Proof. According to the definition of indifferentiability, we have that, in the real world, the differentiator has three honest interfaces (iSch.GEN, iSch.SIGN, iSch.VERIFY) and adversarial interfaces including random oracles $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group **GGM** = (**Label**, **Add**), and ideal cipher **ICM** =(**E**, $\mathbf{E}^{-1}$). Therefore, to complete the proof, we build an efficient simulator $\mathcal{S}$ in the ideal world, such that 1) $\mathcal{S}$ has access to the ideal signature via the adversarial interfaces; 2) $\mathcal{S}$ simulates those seven adversarial interfaces properly. Concretely, in the ideal world, the differentiator $\mathcal{D}$ has three honest interfaces (**Gen**, **Sign**, **Verify**) and seven adversarial interfaces ($\mathcal{S}^{\mathbf{H}_{\text{sk}}}$, $\mathcal{S}^{\mathbf{H}_{\text{seed}}}$, $\mathcal{S}^{\mathbf{H}_{\text{test}}}$, $\mathcal{S}^{\mathbf{Label}}$, $\mathcal{S}^{\mathbf{Add}}$, $\mathcal{S}^{\mathbf{E}}$, $\mathcal{S}^{\mathbf{E}^{-1}}$), and we prove that for any differentiator $\mathcal{D}$, the view in the real world is close to the view in the ideal world. In the following, we illustrate the full description of our simulator and then we give the high-level intuition of our proof strategy.

Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ has the external oracle access to the ideal signature **SIG** =(**Gen**, **Sign**, **Verify**); the simulator $\mathcal{S}$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}^{\mathbf{H}_{sk}}(SK)}$:
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{sk}}$, then return $sk$;
query the external SIG with (Gen, $SK$), and obtain $\widehat{PK}$;
if $\exists(\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{sk}}$ s.t. $PK = \widehat{PK}$, then return $sk$;
$sk \twoheadleftarrow \mathbb{Z}_p$; $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, \widehat{PK})\}$; $\mathbb{T}_{\mathbf{H}_{sk}} \leftarrow \mathbb{T}_{\mathbf{H}_{sk}} \cup \{(SK, sk, \widehat{PK})\}$;
return $sk$.

$\underline{\mathcal{S}^{\mathbf{Label}}(sk)}$:
if $\exists(sk, g_{sk}) \in \mathbb{T}_{\mathbf{Label}}$, then return $g_{sk}$;
$SK \twoheadleftarrow \mathcal{SK}$, query the SIG with (Gen, $SK$), and obtain $\widehat{PK}$;
$\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, \widehat{PK})\}$; $\mathbb{T}_{\mathbf{H}_{sk}} \leftarrow \mathbb{T}_{\mathbf{H}_{sk}} \cup \{(SK, sk, \widehat{PK})\}$;
return $\widehat{PK}$.

$\underline{\mathcal{S}^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$:
If $\nexists(sk_1, g_{sk_1}) \in \mathbb{T}_{\mathbf{Label}}$,
    then $sk_1 \leftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{sk}} \leftarrow \mathbb{T}_{\mathbf{H}_{sk}} \cup \{(\diamond, sk_1, g_{sk_1})\}$,
         $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk_1, g_{sk_1})\}$
If $\nexists(sk_2, g_{sk_2}) \in \mathbb{T}_{\mathbf{Label}}$,
    then $sk_2 \leftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{sk}} \leftarrow \mathbb{T}_{\mathbf{H}_{sk}} \cup \{(\diamond, sk_2, g_{sk_2})\}$,
         $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk_2, g_{sk_2})\}$
$sk \leftarrow sk_1 + sk_2 \bmod p$, return $\mathcal{S}^{\mathbf{Label}}(sk)$.

$\underline{\mathcal{S}^{\mathbf{H}_{seed}}(PK, SK, M, R)}$:
$\widehat{PK} \leftarrow \mathcal{S}^{\mathbf{H}_{sk}}(SK)$;
if $\exists(PK, SK, M, R, seed) \in \mathbb{T}_{\mathbf{H}_{seed}}$, then return $seed$;
query the external SIG with (Sign, $PK, SK, M, R$), and obtain $\widehat{V}$,
if $\exists(\widehat{V}, PK, M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$,
    then return $seed$;
$seed \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{seed}} \leftarrow \mathbb{T}_{\mathbf{H}_{seed}} \cup \{(PK, SK, M, R, seed)\}$,
return $\mathit{seed}$.

$\underline{\mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})}$:
if $\exists (PK, M, \widehat{g}, \mathit{test}) \in \mathbb{T}_{\mathbf{H}_{\text{test}}}$, then return $\mathit{test}$;
$\mathit{test} \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\text{test}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{test}}} \cup \{(PK, M, \widehat{g}, \mathit{test})\}$,
return $\mathit{test}$.

$\underline{\mathcal{S}^{\mathbf{E}}(PK||M, V_1, V_2)}$:
if $\exists (V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $V$;
if $\exists (V, PK||M, V_1, V_2, \mathit{seed}) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $V$;
if $\exists (SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big)$
    then goto Case 1;  //Bad event: secret key collision
if $\nexists (SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$,
    then goto Case 1;  //Bad event: no knowledge of the corresponding secret key
    else $\widehat{SK} \leftarrow SK$; $\widehat{sk} \leftarrow sk$;
$\mathit{seed} \leftarrow V_1 + V_2 \cdot \widehat{sk} \bmod p$;
if $\nexists (\mathit{seed}, g_{\mathit{seed}}) \in \mathbb{T}_{\mathbf{Label}}$,  //Bad event: $\mathbf{Label}(\mathit{seed})$ has never been queried
    then go to Case 1;
if $V_2 \neq \mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, g_{\mathit{seed}})$, then $g_{\mathit{seed}} \leftarrow \mathcal{S}^{\mathbf{Label}}(\mathit{seed})$;
if $\exists (PK, \widehat{SK}, M, R, \mathit{seed}) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$,
    then $\widehat{R} \leftarrow R$; go to Case 3;
    else go to Case 2.  //Semi-bad event: $(V_1, V_2)$ passes verification with abused nonce

Case 1:  //One of the bad events occurs, then responds with random signature value
    $V \twoheadleftarrow \mathbf{\Sigma}$; $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
Case 2:  //Semi-bad event occurs: then responds with a valid signature by sampling a nonce
    $\widehat{R} \twoheadleftarrow \mathcal{R}$; $\mathbb{T}_{\mathbf{H}_{\text{seed}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{seed}}} \cup \{PK, \widehat{SK}, M, \widehat{R}, \mathit{seed}\}$,
    query the external $\mathbf{SIG}$ with $(\mathbf{Sign}, PK, \widehat{SK}, M, \widehat{R})$, and obtain $V$;
    $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
Case 3: query the external $\mathbf{SIG}$ with $(\mathbf{Sign}, PK, \widehat{SK}, M, \widehat{R})$, and obtain $V$;
    $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.

$\underline{\mathcal{S}^{\mathbf{E}^{-1}}(PK||M, V)}$:
if $\exists (V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $(V_1, V_2)$.
if $\exists (V, PK||M, V_1, V_2, \mathit{seed}) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $(V_1, V_2)$.
query the external $\mathbf{SIG}$ with $(\mathbf{Verify}, PK||M, V)$, and obtain $\widehat{\phi}$;
if $\widehat{\phi} = 0$,  //for the invalid signature, respond with random strings
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $(V_1, V_2)$.
if $\exists (SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big)$,
//Bad event: secret key collision, then responds with random strings
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $(V_1, V_2)$.
if $\exists (SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$ or $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$ ,
    then $\widehat{sk} \leftarrow sk$  //identify the secret key ;
    else $\widehat{sk} \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{sk}}} \cup \{(\diamond, \widehat{sk}, PK)\}$; $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(\widehat{sk}, PK)\}$
$\widehat{\mathit{seed}} \twoheadleftarrow \mathbb{Z}_p$;
for each tuple $(PK^*, SK^*, M^*, R^*, \mathit{seed}^*) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$,  //identify the nonce
    query the external $\mathbf{SIG}$ with $(\mathbf{Sign}, PK^*, SK^*, M^*, R^*)$, and obtain $V^*$,
    if $PK^* = PK, M^* = M, V^* = V$, then $\widehat{\mathit{seed}} \leftarrow \mathit{seed}^*$;
$g_{\widehat{\mathit{seed}}} \leftarrow \mathcal{S}^{\mathbf{Label}}(\widehat{\mathit{seed}})$; $\mathit{test} \leftarrow \mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, g_{\widehat{\mathit{seed}}})$;
//compute the Schnorr signature $(V_1, V_2)$ using $\widehat{sk}$, $\widehat{\mathit{seed}}$ and test
$V_1 \leftarrow \widehat{\mathit{seed}} - \widehat{sk} \cdot \mathit{test}$; $V_2 \leftarrow \mathit{test}$; $\mathbb{T}_{\mathbf{E}^{-1}} \leftarrow \mathbb{T}_{\mathbf{E}^{-1}} \cup \{(V, PK, M, V_1, V_2, \mathit{seed})\}$; return $(V_1, V_2)$.
```

We immediately note that, if the differentiator $\mathcal{D}$ makes q queries via the honest and adversarial interfaces, then our simulator $\mathcal{S}$ makes at most q^2 number of queries to the ideal signature $\mathbf{SIG} = (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, which indicates that the simulator above $\mathcal{S}$ is efficient. Next we prove that $\mathcal{S}$ responds to all the queries properly, which implies that the view of $\mathcal{D}$ in the real world is close to the view in the ideal world. To do so, we introduce a sequence of hybrid games, $\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}}, \ldots, \mathcal{G}_{\mathcal{S}_8}^{\mathcal{F}_8,\mathcal{D}}$, and proving that: 1) $\Pr[\mathbf{Real}^{\Pi,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1]$; 2) $\left|\Pr[\mathcal{G}_{\mathcal{S}_i}^{\mathcal{F}_i,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_{i+1}}^{\mathcal{F}_{i+1},\mathcal{D}} = 1]\right| \leq \text{negl}(\lambda)$; 3) $\Pr[\mathcal{G}_{\mathcal{S}_8}^{\mathcal{F}_8,\mathcal{D}} = 1] = \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{SIG},\mathcal{D}}]$, where $\mathcal{F}_i$ and $\mathcal{S}_i$ are the corresponding ideal functionality and simulator in each hybrid game, respectively.

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

Simulator $\mathcal{S}_0$

The simulator $\mathcal{S}_0$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_0$ has the external oracle access to $\mathcal{F}_0$; the simulator $\mathcal{S}_0$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_0^{\mathbf{H}_{\text{sk}}}(SK):}$
$sk \leftarrow \mathbf{H}_{\text{sk}}(SK)$, return $sk$.

$\underline{\mathcal{S}_0^{\mathbf{Label}}(sk):}$
$\mathbf{Label}(sk) \leftarrow \mathbf{H}_{\text{sk}}(SK)$, return $\mathbf{Label}(sk)$.

$\underline{\mathcal{S}_0^{\mathbf{Add}}(g_{sk_1}, g_{sk_2}):}$
$g^* \leftarrow \mathbf{Add}(g_{sk_1}, g_{sk_2})$, return $g^*$.

$\underline{\mathcal{S}_0^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R):}$
$seed \leftarrow \mathbf{H}_{\text{seed}}(PK, SK, M, R)$, return $seed$.

$\underline{\mathcal{S}_0^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g}):}$
$test \leftarrow \mathbf{H}_{\text{test}}(PK, M, g)$, return $test$.

$\underline{\mathcal{S}_0^{\mathbf{E}}(PK||M,\ V_1||V_2):}$
$V \leftarrow \mathbf{E}(PK||M,\ V_1,\ V_2)$, return $V$.

$\underline{\mathcal{S}_0^{\mathbf{E}^{-1}}(PK||M,\ V):}$
$(V_1,\ V_2) \leftarrow \mathbf{E}^{-1}(PK||M,\ V)$, return $(V_1,\ V_2)$.
```

It is straightforward that the view of $\mathcal{D}$ in either $\mathbf{Real}$ or $\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}}$ are identical, thus $\Pr[\mathbf{Real}^{\Pi,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0,\mathcal{D}} = 1]$. Then we give the next simulator $\mathcal{S}_1$,

Simulator $\mathcal{S}_1$

The simulator $\mathcal{S}_1$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_1$ has the external oracle access to $\mathcal{F}_1$; the simulator $\mathcal{S}_1$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_1^{\mathbf{H}_{\text{sk}}}(SK):}$
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$, then return $sk$;
query the external $\mathcal{F}_1$ with ($\mathbf{Gen}$, $SK$), and obtain $\widehat{PK}$;
if $\exists(\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$ s.t. $PK = \widehat{PK}$, then return $sk$;
$sk \leftarrow \mathbf{H}_{\text{sk}}(SK)$, $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, \widehat{PK})\}$; $\mathbb{T}_{\mathbf{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{sk}}} \cup \{(SK, sk, \widehat{PK})\}$;
```

```
return $sk$.

$\underline{\mathcal{S}_1^{\mathbf{Label}}(sk):}$
if $\exists(sk, g_{sk}) \in \mathbb{T}_{\mathbf{Label}}$, then return $g_{sk}$;
$g_{sk} \leftarrow \mathbf{Label}(sk)$, $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, g_{sk})\}$,
return $g_{sk}$.

$\underline{\mathcal{S}_1^{\mathbf{Add}}(g_{sk_1}, g_{sk_2}):}$
If $\nexists(sk_1, g_{sk_1}) \in \mathbb{T}_{\mathbf{Label}}$, then return $\mathbf{Add}(g_{sk_1}, g_{sk_2})$;
If $\nexists(sk_2, g_{sk_2}) \in \mathbb{T}_{\mathbf{Label}}$, then return $\mathbf{Add}(g_{sk_1}, g_{sk_2})$;
$sk \leftarrow sk_1 + sk_2 \bmod p$, return $\mathcal{S}_1^{\mathbf{Label}}(sk)$.

$\underline{\mathcal{S}_1^{\mathbf{H}_{\mathrm{seed}}}(PK, SK, M, R):}$
$\widehat{PK} \leftarrow \mathbf{Gen}(SK)$;
if $\exists(PK, SK, M, R, seed) \in \mathbb{T}_{\mathbf{H}_{\mathrm{seed}}}$, then return $seed$;
query the external $\mathcal{F}_1$ with $(\mathbf{Sign}, PK, SK, M, R)$, and obtain $\widehat{V}$,
if $\exists(\widehat{V}, PK, M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $seed$;
$seed \leftarrow \mathbf{H}_{\mathrm{seed}}(PK, SK, M, R)$, $\mathbb{T}_{\mathbf{H}_{\mathrm{seed}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{seed}}} \cup \{(PK, SK, M, R, seed)\}$,
return $seed$.

$\underline{\mathcal{S}_1^{\mathbf{H}_{\mathrm{test}}}(PK, M, \widehat{g}):}$
if $\exists(PK, M, \widehat{g}, test) \in \mathbb{T}_{\mathbf{H}_{\mathrm{test}}}$, then return $test$;
$test \leftarrow \mathbf{H}_{\mathrm{test}}(PK, M, \widehat{g})$, $\mathbb{T}_{\mathbf{H}_{\mathrm{test}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{test}}} \cup \{(PK, M, \widehat{g}, test)\}$,
return $test$.

$\underline{\mathcal{S}_1^{\mathbf{E}}(PK||M, V_1||V_2):}$
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $V$;
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $V$;
if $\exists(SK \neq SK')$ s.t. $\big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}\big) \wedge \big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}\big)$
    then goto Case 1; //Bad event: secret key collision
if $\nexists(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}$,
    then goto Case 1; //Bad event: no knowledge of the corresponding secret key
    else $\widehat{SK} \leftarrow SK$; $\widehat{sk} \leftarrow sk$;
$seed \leftarrow V_1 + V_2 \cdot \widehat{sk} \bmod p$;
if $\nexists(seed, g_{seed}) \in \mathbb{T}_{\mathbf{Label}}$, then $g_{seed} \leftarrow \mathcal{S}_1^{\mathbf{Label}}(seed)$;
if $V_2 \neq \mathcal{S}^{\mathbf{H}_{\mathrm{test}}}(PK, M, g_{seed})$, //Bad event: $(V_1, V_2)$ fails the verification phase
    then go to Case 1;
if $\exists(PK, \widehat{SK}, M, R, seed) \in \mathbb{T}_{\mathbf{H}_{\mathrm{seed}}}$,
    then $\widehat{R} \leftarrow R$; go to Case 3;
    else go to Case 2. //Semi-bad event: $(V_1, V_2)$ passes verification with abused nonce

Case 1: //One of the bad events occurs, then responds by calling $\mathbf{E}$
        $V \leftarrow \mathbf{E}(PK||M, V_1||V_2)$; $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
Case 2: //Semi-bad event occurs: then responds by calling $\mathbf{E}$
        $V \leftarrow \mathbf{E}(PK||M, V_1||V_2)$; $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
        $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
Case 3: query the external $\mathcal{F}_1$ with $(\mathbf{Sign}, PK, \widehat{SK}, M, \widehat{R})$, and obtain $V$;
        $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.

$\underline{\mathcal{S}_1^{\mathbf{E}^{-1}}(PK||M, V):}$
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $(V_1, V_2)$.
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $(V_1, V_2)$.
query the external $\mathcal{F}_1$ with $(\mathbf{Verify}, PK||M, V)$, and obtain $\widehat{\phi}$;
if $\widehat{\phi} = 0$, //for the invalid signature, respond by calling $\mathbf{E}^{-1}$
```

```
    then $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V)$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;    return $(V_1, V_2)$.
if $\exists(SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big)$,
//Bad event: secret key collision, then responds by calling $\mathbf{E}^{-1}$
    then $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V)$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
if $\exists(SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$ or $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$ ,
    then $\widehat{sk} \leftarrow sk$  //identify the secret key ;
    else $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V)$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
$\widehat{seed} \leftarrow \diamond$;
for each tuple $(PK^*, SK^*, M^*, R^*, seed^*) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$,  //identify the nonce
    query the external $\mathcal{F}_1$ with $(\mathbf{Sign}, PK^*, SK^*, M^*, R^*)$, and obtain $V^*$,
    if $PK^* = PK, M^* = M, V^* = V$,
      then $\widehat{seed} \leftarrow seed^*$;
if $\widehat{seed} = \diamond$,
    then $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V)$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $(V_1, V_2)$.
$g_{\widehat{seed}} \leftarrow \mathcal{S}^{\mathbf{Label}}(\widehat{seed})$; $test \leftarrow \mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, g_{\widehat{seed}})$;
//compute the Schnorr signature $(V_1, V_2)$ using $\widehat{sk}$, $\widehat{seed}$ and $test$
$V_1 \leftarrow \widehat{seed} - \widehat{sk} \cdot test$; $V_2 \leftarrow test$; $\mathbb{T}_{\mathbf{E}^{-1}} \leftarrow \mathbb{T}_{\mathbf{E}^{-1}} \cup \{(V, PK, M, V_1, V_2, seed)\}$; return $(V_1, V_2)$.
```

Comparing to $\mathcal{S}_0$, the simulator $\mathcal{S}_1$ keeps several tables. When $\mathcal{S}_1$ responds to a query, it first checks the tables, if the proper response is recorded in one of the tables, then $\mathcal{S}_1$ responds to the query using the table, else $\mathcal{S}_1$ responds to the query by using its internal copies of the oracles. In the following, we prove that, for any query, the response of $\mathcal{S}_0$ and $\mathcal{S}_1$ is identical with high probability, which straightforwardly refers to that

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

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

$$\mathcal{S}_0^{\mathbf{H}_{\text{sk}}}(SK) = \mathcal{S}_1^{\mathbf{H}_{\text{sk}}}(SK).$$

Applying exactly the same analysis, it's straightforward that $\mathcal{S}_0^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R) = \mathcal{S}_1^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$, $\mathcal{S}_0^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g}) = \mathcal{S}_1^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$, $\mathcal{S}_0^{\mathbf{Label}}(sk) = \mathcal{S}_1^{\mathbf{Label}}(sk)$.

Thus it suffices to show that for $\mathbf{E}, \mathbf{E}^{-1}$, the equivalence also holds. In Game 0, we have that

$$\mathcal{S}_0^{\mathbf{E}}(PK||M, V_1, V_2) = \mathbf{E}(PK||M, V_1||V_2),$$

and in Game 1, $\mathcal{S}_1$ responds to this query in the following cases:

1. using its table $\mathbb{T}_{\mathbf{E}}$;
2. using its table $\mathbb{T}_{\mathbf{E}^{-1}}$;
3. calling $\mathbf{E}(PK||M, V_1||V_2)$;
4. responds with $\Pi.\textsc{Sign}(PK, \widehat{SK}, M, R)$.

Due to the analysis above, we immediately observe that the tuples recorded in tables are consistent with the oracles, and thus it suffices to *only* analyze the last case. By the description of $\mathcal{S}_1$, we have that $\mathcal{S}_1^{\mathbf{E}}(PK||M,\, V_1,\, V_2) = \Pi.\textsc{Sign}(PK, \widehat{SK}, M, R)$ if and only if the following conditions hold (case 3 occurs):

- there is a tuple $(\widehat{SK}, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$;
- there is a tuple $(PK, \widehat{SK}, M, R, \mathit{seed}) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$;
- there is a tuple $(\mathit{seed}, g_{\mathit{seed}}) \in \mathbb{T}_{\mathbf{Label}}$;
- there is a tuple $(PK, M, g_{\mathit{seed}}, \mathit{test}) \in \mathbb{T}_{\mathbf{H}_{\text{test}}}$;
- $V_1 = \mathit{seed} - sk \cdot \mathit{test},\, V_2 = \mathit{test}$.

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

$$\Pi.\textsc{Sign}(PK, \widehat{SK}, M, R) = \mathbf{E}(PK||M,\, V_1||\, V_2).$$

Thus we have that

$$\mathcal{S}_0^{\mathbf{E}}(PK||M,\, V_1||\, V_2) = \mathcal{S}_1^{\mathbf{E}}(PK||M,\, V_1||\, V_2).$$

And for $\mathbf{E}^{-1}$, in Game 0, $\mathcal{S}_0^{\mathbf{E}^{-1}}(PK||M,\, V) = \mathbf{E}^{-1}(PK||M,\, V)$. While, in Game 1, $\mathcal{S}_1$ responds to this query in the following cases:

1. using its table $\mathbb{T}_{\mathbf{E}}$;
2. using its table $\mathbb{T}_{\mathbf{E}^{-1}}$;
3. calling $\mathbf{E}^{-1}(\cdot, \cdot)$;
4. responds with $(V_1,\, V_2)$.

Applying the same analysis above, we have that, if $\mathcal{S}_1$ responds to the query by the first three cases, then it's apparent that $\mathcal{S}_0^{\mathbf{E}^{-1}}(PK||M,\, V) = \mathcal{S}_1^{\mathbf{E}^{-1}}(PK||M,\, V)$. And next we analyze the last case. Essentially, $\mathcal{S}_1^{\mathbf{E}^{-1}}(PK||M,\, V) = (V_1,\, V_2)$ if and only if the following conditions hold:

- $\Pi.\textsc{Verify}(PK||M,\, V) = 1$;
- there exist no $(SK, sk, PK) \neq (SK', sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$;
- there is a tuple $(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$;
- there is a tuple $(PK, SK, M, R, \mathit{seed}) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$ such that $\Pi.\textsc{Sign}(PK, SK, M, R) = V$;

Then $\mathcal{S}_1$ responds to the query with

$$V_2 = \mathcal{S}_1^{\mathbf{H}_{\text{test}}}(PK, M, \mathcal{S}_1^{\mathbf{Label}}(\mathit{seed})),\, V_1 = \mathit{seed} - sk \cdot V_2.$$

Moreover, we have that $V = \Pi.\textsc{Sign}(PK, SK, M, R) = \mathbf{E}(PK||M,\, V_1||\, V_2)$, which refers to

$$\mathcal{S}_0^{\mathbf{E}^{-1}}(PK||M,\, V) = \mathcal{S}_1^{\mathbf{E}^{-1}}(PK||M,\, V).$$

Combing together, it's apparent that

$$\Pr[\mathcal{G}_{\mathcal{S}_0}^{\mathcal{F}_0, \mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_1}^{\mathcal{F}_1, \mathcal{D}} = 1].$$

Moreover, we note that the strategies for responding to all queries of $\mathcal{S}_1$ are same as the ones in our final simulator $\mathcal{S}$, except that

1. $\mathcal{S}_1$ keeps several internal copies for the oracles and $\mathcal{S}$ keeps none;
2. $\mathcal{S}_1$ responds to the queries by using the tables (if proper tuples stored) or the internal copies, while $\mathcal{S}$ only uses its tables and external oracle access.

Therefore, in the next hybrid games, we erase the internal copies for the oracles one by one and finally reach out to $\mathcal{S}$. Due to space limit, we give the rest proof in Appendix B.

□

4 Indifferentiable Boneh-Boyen Signature

In this section, we show how to "lift" the Boneh-Boyen signature to achieve indifferentiability.

4.1 Boneh-Boyen signature

To do that, we first recall the Boneh-Boyen signature scheme $\mathsf{BB} = (\mathsf{BB.}\textsc{Gen}, \mathsf{BB.}\textsc{Sign}, \mathsf{BB.}\textsc{Verify})$:

- Key generation $(PK, SK) \twoheadleftarrow \mathsf{BB.}\textsc{Gen}(1^\lambda)$:
  $(\mathbb{G}_1, \mathbb{G}_2, e, p, g_1, g_2) \twoheadleftarrow \mathsf{GroupGen}(1^\lambda)$; $x, y \twoheadleftarrow \mathbb{Z}_p$; $X \leftarrow g_1^x$; $Y \leftarrow g_1^y$;
  $SK \leftarrow (x, y)$; $PK \leftarrow (X, Y)$; $\mathsf{params} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, e, p, g_1, g_2, e(g_1, g_2))$;
- Signing $V \twoheadleftarrow \mathsf{BB.}\textsc{Sign}(\mathsf{params}, SK, M)$:
  Parse SK into (x, y); $r \twoheadleftarrow \mathbb{Z}_p$; $V \leftarrow (g_2^{\frac{1}{x+M+y\cdot r}}, r)$;
- Verification $\phi \leftarrow \mathsf{BB.}\textsc{Verify}(\mathsf{params}, PK, M, V)$:
  Parse PK into (X, Y), parse V into (V_1, V_2);
  outputs $\phi = 1$ if and only if $e(X \cdot g_1^M \cdot Y^{V_2}, V_1) = e(g_1, g_2)$.

Correctness of Boneh-Boyen signature holds straightforwardly, in fact, $X \cdot g_1^M \cdot Y^{V_2} = g_1^{x+M+y\cdot r}$, which refers to that $e(X \cdot g_1^M \cdot Y^{V_2}, V_1) = e(g_1^{x+M+y\cdot r}, g_2^{\frac{1}{x+M+y\cdot r}}) = e(g_1, g_2)$.

The security has been proven in [BB04]: Boneh-Boyen signature scheme can achieve strong unforgeability in the standard model, under the q-strong Diffie-Hellman assumption.

4.2 Indifferentiable Boneh-Boyen Signature

In this part, we give the description of the indifferentiable Boneh-Boyen signature, and here are the building blocks:

- $\mathbf{H}_{\text{sk-one}} : \{0,1\}^* \to [p]$ is a random oracle model;
- $\mathbf{H}_{\text{sk-two}} : \{0,1\}^* \to [p]$ is a random oracle model;
- $\mathbf{H}_{\text{seed}} : \{0,1\}^* \to [p]$ is a random oracle model;
- $\mathbf{GBM} = (\mathbf{Label_1}, \mathbf{Label_2}, \mathbf{Label_T}, \mathbf{Add_1}, \mathbf{Add_2}, \mathbf{Add_T}, \mathbf{Mult})$ is a generic bilinear group model.
- $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$ is a ideal cipher, where $\mathbf{E} : \{0,1\}^k \times \{0,1\}^t \to \{0,1\}^t$, where $k = \log|\mathcal{PK}| + \log|\mathcal{M}|$ and $t = 2\lceil \log p \rceil$ and $\mathbf{E}^{-1}$ is its inverse.

The following is the indifferentiable BB signature, where the public parameter $\mathsf{params} \leftarrow (\mathbf{H}_{\text{sk-one}}, \mathbf{H}_{\text{sk-two}}, \mathbf{H}_{\text{seed}}, \mathbf{GBM}, \mathbf{ICM})$.

$i\mathsf{BB} = (i\mathsf{BB}.\textsc{Gen}, i\mathsf{BB}.\textsc{Sign}, i\mathsf{BB}.\textsc{Verify})$

```
$PK \leftarrow i\mathsf{BB}.\textsc{Gen}(SK)$:
On input, signing key $SK$, compute the verification key $PK$, as follows:
    1. parse $SK$ into $(SK_1, SK_2)$;
    2. $sk_1 \leftarrow \mathbf{H}_{\rm sk}(SK_1)$; $sk_2 \leftarrow \mathbf{H}_{\rm sk}(SK_2)$;
       $PK_1 \leftarrow \mathbf{Label_1}(sk_1)$; $PK_2 \leftarrow \mathbf{Label_1}(sk_2)$; $PK \leftarrow (PK_1, PK_2)$;
    3. return $PK$.

$V \leftarrow i\mathsf{BB}.\textsc{Sign}(PK, SK, M, R)$:
On input, verification key $PK$, signing key $SK$, message $M$, nonce $R$ compute the signature $V$, as follows:
    If $PK = i\mathsf{BB}.\textsc{Gen}(SK)$
        then $seed \leftarrow \mathbf{H}_{\rm seed}(PK, SK, M, R)$;
            parse $SK$ into $(SK_1, SK_2)$;
            $v_1 \leftarrow \mathbf{Label_2}(\frac{1}{\mathbf{H}_{\rm sk}(SK_1)+M+\mathbf{H}_{\rm sk}(SK_2)\cdot seed}) = \mathbf{Label_2}(\frac{1}{sk_1+M+sk_2\cdot seed})$;
           $v_2 \leftarrow seed$;
            output the signature $V \leftarrow \mathbf{E}\Big(PK||M,\ v_1||v_2\Big)$.
    else output the signature $V \leftarrow \perp$; //aborts if $PK$ is invalid

$\phi \leftarrow i\mathsf{BB}.\textsc{Verify}(PK, M, V)$:
On input, verification key $PK$, message $M$ and signature $V$, operate as follows:
    1. "unpack" the signature $(v_1, v_2) \leftarrow \mathbf{E}^{-1}(PK||M,\ V)$;
    2. parse the public key $(PK_1, PK_2) \leftarrow PK$;
    3. compute the corresponding group element
       $g^* = \mathbf{Add_1}(PK_1, PK_2^{v_2}, \mathbf{Label_1}(M)) = \mathbf{Label_1}(sk_1 + M + sk_2 \cdot v_2)$;
    4. output $\phi \leftarrow 1$ if $\mathbf{Mult}(\mathbf{Label_1}(1), \mathbf{Label_2}(1)) = \mathbf{Mult}\Big(g^*, v_1\Big)$;
       output $\phi \leftarrow 0$, otherwise.
```

In real-world, we can instantiate the generic group model $\mathbf{GBM}$ by applying the elliptic curves in [BLS01].

Remark 4.1 (Fallback security). *Our construction $i\mathsf{BB}$ here enjoys a nice feature:*

- *when generic group model $\mathbf{GBM}$ does exist, our construction can achieve very strong security guarantee, i.e., indifferentiable from the ideal signatures;*
- *when generic group model $\mathbf{GBM}$ does not exist and boils down to ordinary bilinear group, our construction here can still be able to achieve the same security guarantee as that has been achieved by the original Boneh-Boyen signature; that is, our construction can achieve strong unforgeability in the standard, assuming the q-strong Diffie-Hellman problem is hard over the bilinear groups, and $(\mathbf{E}, \mathbf{E}^{-1})$ is a keyed permutation only requiring correctness.*

4.3 Security

The correctness of the our indifferentiable BB-signature holds trivially, and this part we prove that it is indifferentiable from an ideal signature. More concretely, we have the following theorem.

Theorem 4.2. *$i\mathsf{BB} = (i\mathsf{BB}.\textsc{Gen}, i\mathsf{BB}.\textsc{Sign}, i\mathsf{BB}.\textsc{Verify})$ is indifferentiable from the ideal signature $\mathbf{SIG} = (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, in the model for, random oracles $\mathbf{H}_{\text{sk-one}}$, $\mathbf{H}_{\text{sk-two}}$, $\mathbf{H}_{\rm seed}$, generic bilinear group $\mathbf{GBM} = (\mathbf{Label_1}, \mathbf{Label_2}, \mathbf{Label_T}, \mathbf{Add_1}, \mathbf{Add_2}, \mathbf{Add_T}, \mathbf{Mult})$, and ideal cipher $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathsf{Adv}^{\mathsf{indiff}}_{i\mathsf{BB},\mathbf{SIG},\mathcal{S}}(\mathcal{D}) \leq O(\frac{q^2}{p}).$$

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

The proof is similar to the one in Section 3 and we omit it here.

4.4 Additional constructions

In this part, we build two additional indifferentiable signature schemes which achieve better efficiency. Recall that, in [BB04] Boneh and Boyen also propose alternative scheme that yields better efficiency, whose signature only consists of a single group element. The trade-off is that the scheme, denoted as wBB, only achieves weak unforgeability. In the following, we show how to lift wBB to achieve indifferentiability in the generic bilinear groups model, by introducing almost no extra efficiency cost. We remark that, the ideas here can also be applied to the signature scheme by Zhang et al [ZSS04].

4.4.1 Weak Boneh-Boyen signature

We first recall the weak Boneh-Boyen signature scheme, denoted as $\mathsf{wBB} = (\mathsf{wBB}.\textsc{Gen}, \mathsf{wBB}.\textsc{Sign}, \mathsf{wBB}.\textsc{Verify})$:

- Key generation $(PK, SK) \twoheadleftarrow \mathsf{wBB}.\textsc{Gen}(1^\lambda)$:
  $(\mathbb{G}_1, \mathbb{G}_2, e, p, g_1, g_2) \twoheadleftarrow \mathsf{GroupGen}(1^\lambda)$; $x \twoheadleftarrow \mathbb{Z}_p$; $X \leftarrow g_1^x$;
  $SK \leftarrow x$; $PK \leftarrow X$; $\mathsf{params} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, e, p, g_1, g_2, e(g_1, g_2))$;
- Signing $V \twoheadleftarrow \mathsf{wBB}.\textsc{Sign}(PK, SK, M)$:
  $V \leftarrow g_2^{\frac{1}{M+SK}}$;
- Verification $\phi \leftarrow \mathsf{wBB}.\textsc{Verify}(\mathsf{params}, PK, M, V)$:
  $\phi = 1$ if and only if $e(g_1^M \cdot PK, V) = e(g_1, g_2)$.

Correctness holds. In fact, $g_1^M \cdot PK = g_1^{M+SK}$, referring to $e(g_1^M \cdot PK, V) = e(g_1^{M+SK}, g_2^{\frac{1}{M+SK}}) = e(g_1, g_2)$.

4.4.2 Indifferentiable wBB

Next, we describe the indifferentiable weak Boneh-Boyen signature, and here are the building blocks:

- $\mathbf{H}_{\text{sk}} : \{0,1\}^* \to [p]$ is a random oracle model;
- $\mathbf{H}_{\text{msg}} : \{0,1\}^* \to [p]$ is a random oracle model;
- $\mathbf{GBM} = (\mathbf{Label_1}, \mathbf{Label_2}, \mathbf{Label_T}, \mathbf{Add_1}, \mathbf{Add_2}, \mathbf{Add_T}, \mathbf{Mult})$ is a generic bilinear group.
- $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$ is a ideal cipher, where $\mathbf{E} : \{0,1\}^k \times \{0,1\}^t \to \{0,1\}^t$, where $k = \log|\mathcal{PK}| + \log|\mathcal{M}|$ and $t = \lceil \log p \rceil$, and $\mathbf{E}^{-1}$ is its inverse.

The following is the construction of i-wBB_1 along with the public parameters $\mathsf{params} = (\mathbf{H}_{\text{sk}}, \mathbf{GBM}, \mathbf{ICM})$:

i-$\mathsf{wBB}_1 = (i$-$\mathsf{wBB}_1.\textbf{Gen}, i$-$\mathsf{wBB}_1.\textbf{Sign}, i$-$\mathsf{wBB}_1.\textbf{Verify})$

$PK \leftarrow i$-$\mathsf{wBB}_1.\textsc{Gen}(SK)$:
On input, signing key SK, compute the verification key PK, as follows:

```
1. $sk \leftarrow \mathbf{H}_{\text{sk}}(SK)$; $PK \leftarrow \mathbf{Label_1}(sk)$;
2. return $PK$.
```

```
$V \leftarrow i\text{-wBB}_1.\textsc{Sign}(PK, SK, M)$:
On input, verification key $PK$, signing key $SK$, message $M$, compute the signature $V$, as follows:
    If $PK = \textsc{Gen}(SK)$
        then $v \leftarrow \mathbf{Label_2}(\frac{1}{M+\mathbf{H}_{\text{sk}}(SK)})$;
            output the signature $V \leftarrow \mathbf{E}\Big(PK||M,\ v\Big)$;
        else output the signature $V \leftarrow \perp$;
$\phi \leftarrow i\text{-wBB}_1.\textsc{Verify}(PK, M, V)$:
On input, verification key $PK$, message $M$ and signature $V$, operate as follows:
    1. "unpack" the signature $v \leftarrow \mathbf{E}^{-1}(PK||M,\ V)$;
    2. compute the corresponding group element:
           $g^* = PK \cdot \mathbf{Label_1}(M) = \mathbf{Label_1}(\mathbf{H}_{\text{sk}}(SK) + M)$
    3. output $\phi \leftarrow 1$ if $\mathbf{Mult}(\mathbf{Label_1}(1), \mathbf{Label_2}(1)) = \mathbf{Mult}(g^*, v)$,
       output $\phi \leftarrow 0$ otherwise.
```

Next, we make use of the additional random oracle $\mathbf{H}_{\text{msg}}$ to build a variant indifferentiable weak Boneh-Boyen signature scheme i-wBB$_2$:

$i\text{-wBB}_2 = (i\text{-wBB}_2.\textbf{Gen}, i\text{-wBB}_2.\textbf{Sign}, i\text{-wBB}_2.\textbf{Verify})$

```
$i\text{-wBB}_2.\textsc{Gen}(SK)$: = $i\text{-wBB}_1.\textsc{Gen}(SK)$
$V \leftarrow i\text{-wBB}_2.\textsc{Sign}(PK, SK, M)$:
On input, verification key $PK$, signing key $SK$, message $M$, compute the signature $V$, as follows:
    If $PK = \textsc{Gen}(SK)$
        then $v \leftarrow \mathbf{Label_2}(\frac{1}{\mathbf{H}_{\text{msg}}(M)+\mathbf{H}_{\text{sk}}(SK)})$;
            output the signature $V \leftarrow \mathbf{E}\Big(PK||M,\ v\Big)$;
        else output the signature $V \leftarrow \perp$;
$\phi \leftarrow i\text{-wBB}_2.\textsc{Verify}(PK, M, V)$:
On input, verification key $PK$, message $M$ and signature $V$, operate as follows:
    1. "unpack" the signature $v \leftarrow \mathbf{E}^{-1}(PK||M,\ V)$;
    2. compute the corresponding group element:
           $g^* = PK \cdot \mathbf{Label_1}(M) = \mathbf{Label_1}(\mathbf{H}_{\text{sk}}(SK) + \mathbf{H}_{\text{msg}}(M))$
    3. output $\phi \leftarrow 1$ if $\mathbf{Mult}(\mathbf{Label_1}(1), \mathbf{Label_2}(1)) = \mathbf{Mult}(g^*, v)$,
       output $\phi \leftarrow 0$ otherwise.
```

Remark 4.3 (Fallback security). *Our construction i-wBB$_2$ here enjoys a nice feature:*

- *when generic group model* $\mathbf{GBM}$ *does exist, our construction can achieve very strong security guarantee, i.e., indifferentiable from the ideal signatures;*
- *when generic group model* $\mathbf{GBM}$ *does not exist and boils down to ordinary bilinear group, our construction here can still be able to achieve the same security guarantee as that has been achieved by the original Boneh-Boyen signature; that is, our construction can achieve unforgeability in the random oracle model, assuming the q-strong Diffie-Hellman problem is hard over the bilinear groups, and* $(\mathbf{E}, \mathbf{E}^{-1})$ *is a keyed permutation only requiring correctness.*

4.4.3 Security

The correctness of the our indifferentiable weak BB-signature schemes hold trivially, and next we show both of them are indifferentiable from an ideal signature.

Theorem 4.4. *i-wBB_1 and i-wBB_2 are indifferentiable from the ideal signature $\mathbf{SIG} = (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$, in the model for, random oracles $\mathbf{H}_{\mathrm{sk}}$, $\mathbf{H}_{\mathrm{msg}}$, generic bilinear group $\mathbf{GBM} = (\mathbf{Label_1}, \mathbf{Label_2}, \mathbf{Label_T}, \mathbf{Add_1}, \mathbf{Add_2}, \mathbf{Add_T}, \mathbf{Mult})$, and ideal cipher $\mathbf{ICM} = (\mathbf{E}, \mathbf{E}^{-1})$. More precisely, there exists a simulator $\mathcal{S}$ such that for all q-query differentiator $\mathcal{D}$, we have*

$$\mathsf{Adv}^{\mathsf{indiff}}_{i\text{-}\mathsf{wBB}_1,\mathbf{SIG},\mathcal{S}}(\mathcal{D}) \le O(\frac{q^2}{p}), \mathsf{Adv}^{\mathsf{indiff}}_{i\text{-}\mathsf{wBB}_2,\mathbf{SIG},\mathcal{S}}(\mathcal{D}) \le O(\frac{q^2}{p}).$$

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

5 Related work

Digital Signatures (originally proposed by Diffie and Hellman [DH76]) has widely used. The classical game-based security, also known as unforgeability, was formulated by Goldwasser et al. [GMR88]. In the Universal Composability (UC) framework, digital signature has been rigorously studied, and ideal functionality for signature has been formulated by Canetti [Can04]; there, a thorough investigation between the correspondence of the game-based security formulation [GMR88] and the simulation-based ideal functionality for signature, has been performed. We also note that, the formulation of ideal functionality for signature is non-trivial, and a first attempt of defining the ideal functionality [Can00] has shown not be able to capture the consistency property, as pointed out in [BH04].

Very recently, in the indifferentiability framework [MRH04], digital signature has been studied [ZZ21b]. There, an ideal (one-time) signature model using query-response formulation style as in [RSS11], has been presented. We remark also that, a first attempt of defining ideal signature model [ZZ20], has later been found not complete; see [ZZ21b]. The first construction that is indifferentiable from ideal signature, has been demonstrated in [ZZ21b]; this construction is tree based in the random oracle model. In contrast, the constructions in this paper are much more efficient than that in [ZZ21b], but at the price of using a strictly stronger idealized model, the generic (bilinear) group model.

Finally, we point out here that, the signature that studied in the UC framework and the signature in the indifferentiability framework are fundamentally different; in the UC framework, it has been shown by Canetti [Can04] that a protocol based on the game-based unforgeability (along with additional consistency property and completeness) can be proven to realize the ideal functionality for signature. In contrast, in the indifferentiability framework, the ideal signature is *strictly stronger* than the game-based security for digital signature.

Acknowledgement

We thank Mark Zhandry for the insightful discussion on the early version of this work.

References

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

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

[BH04] Michael Backes and Dennis Hofheinz. How to break and repair a universally composable signature functionality. In Kan Zhang and Yuliang Zheng, editors, *Information Security, 7th International Conference, ISC 2004, Palo Alto, CA, USA, September 27-29, 2004, Proceedings*, volume 3225 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2004.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

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

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[Can04] Ran Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*, page 219. IEEE Computer Society, 2004.

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

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.

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

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

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

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

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

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

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

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

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

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

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In *IMA International Conference on Cryptography and Coding*, pages 1–12. Springer, 2005.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *ICS 2011*, pages 1–21. Tsinghua University Press, January 2011.

[MR16] Ueli Maurer and Renato Renner. From indifferentiability to constructive cryptography (and back). In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 3–24. Springer, Heidelberg, October / November 2016.

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

[NIS94] FIPS 186-4 Digital Signature Standard (DSS). 1994. https://csrc.nist.gov/publications/detail/fips/186/4/final.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *2001 IEEE Symposium on Security and Privacy*, pages 184–200. IEEE Computer Society Press, May 2001.

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

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[ZSS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 277–290. Springer, Heidelberg, March 2004.

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

[ZZ21a] Mark Zhandry and Cong Zhang. The relationship between idealized models under computationally bounded adversaries. 2021. https://eprint.iacr.org/2021/240.

[ZZ21b] Cong Zhang and Hong-Sheng Zhou. From random oracles to ideal signatures, and back. 2021. https://eprint.iacr.org/2021/566.

A Preliminaries: Additional Material

A.1 Idealized model for generic bilinear group

Generic Bilinear Group Model (GBM) [BB04] . For our purposes, a bilinear map consists of three sets S_1, S_2, S_T of prime size p, each endowed with a group structure. Additionally, there is an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that $e(g_1^{z_1}, g_2^{z_2}) = e(g_1, g_2)^{z_1 z_2}$ for some fixed generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.

The Generic Bilinear Group Model is an idealized model which assumes the existence of random embeddings from $\mathbb{Z}_p$ into the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Concretely, a generic bilinear group is a tuple $\mathbf{GBM} = (\mathbf{Label_1}, \mathbf{Label_2}, \mathbf{Label_T}, \mathbf{Add_1}, \mathbf{Add_2}, \mathbf{Add_T}, \mathbf{Mult})$. Here, $\mathbf{Label}_i$ and $\mathbf{Add}_i$, for $i \in \{\mathbf{1}, \mathbf{2}, \mathbf{T}\}$ are as in the generic group case, and $\mathbf{Mult}$ is then defined as $\mathbf{Mult}(\mathbf{Label_1}(z_1), \mathbf{Label_2}(z_2)) = \mathbf{Label_T}(z_1 \cdot z_2)$.

The functionality $\mathbf{GBM} = (\mathbf{GBM}.hon, \mathbf{GBM}.adv)$, as in Figure 5, implements a generic bilinear group and thus defines the idealized model for generic bilinear group.

Remark A.1. *In our paper, we will use a slightly strengthened version of the generic bilinear-group model (GBM). Concretely, let S_{range} be a public known subset of S_1 such that $|S_{\mathsf{range}}| = p$, and the element in S_{range} can be recognized efficiently, which means that, given an $x \in S_1$, one can efficiently tell whether $x \in S_{\mathsf{range}}$ or not.*

In the strengthened GBM, now $\mathbf{Label}_1$ is a "labeling" function that is a random injection from $\mathbb{Z}_p$ to S_{range}. The advantage of our new model is that, comparing to Shoup's model, in our new model, given a string $x \in S_1$, we can easily tell whether x is a valid labeling or not.

A.2 Idealized model for ideal cipher

Functionality $\mathbf{ICM} = (\mathbf{ICM}.hon, \mathbf{ICM}.adv)$, shown in Figure 6, implements an ideal cipher and will give rise to the ideal-cipher model. Note that, two "adversarial" interfaces, $\mathbf{E}.adv$ and $\mathbf{E}^{-1}.adv$ for capturing the encryption and the inverse, respectively, are defined so that the adversaries can access the functionality; and two "honest" interfaces, $\mathbf{E}.hon$ and $\mathbf{E}^{-1}.hon$ are defined for all other procedures to access the functionality.

A.3 Composition Theorem.

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

Theorem A.2. *Let $\mathcal{F}_1$, $\mathcal{F}_2$ be two functionalities with compatible honest interfaces. Let $\mathcal{A}$ be an adversary with one oracle. Let $\mathcal{S}$ be a simulator that exports the same interface as $\mathcal{F}_1.adv$. Then there exist adversary $\mathcal{B}$ and distinguisher $\mathcal{D}$ such that for all values y*

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

Moreover

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

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

```
procedure GBM.hon:
INTERFACE Label_1.hon(x):
If $x \notin \mathbb{Z}_p$
    then return $\perp$;
    else if $\mathbb{T}_1[x] = \perp$ then (re)sample $\mathbb{T}_1[x] \overset{\text{re}}{\twoheadleftarrow} S_1$;
return $\mathbb{T}[x]$.

INTERFACE Label_2.hon(x):
If $x \notin \mathbb{Z}_p$
    then return $\perp$;
    else if $\mathbb{T}_2[x] = \perp$ then (re)sample $\mathbb{T}_2[x] \overset{\text{re}}{\twoheadleftarrow} S_2$;
return $\mathbb{T}_2[x]$.

INTERFACE Label_T.hon(x):
If $x \notin \mathbb{Z}_p$
    then return $\perp$;
    else if $\mathbb{T}_T[x] = \perp$ then (re)sample $\mathbb{T}_T[x] \overset{\text{re}}{\twoheadleftarrow} S_T$;
return $\mathbb{T}_T[x]$.

INTERFACE Add_1.hon(s_1, s_2):
If $\exists x_1, x_2 \in \mathbb{Z}_p$ so that
    $\mathbf{Label_1}.hon(x_1) = s_1$ and $\mathbf{Label_1}.hon(x_2) = s_2$
    then $x_3 \leftarrow x_1 + x_2 \bmod p$;
        return $\mathbf{Label_1}.hon(x_3)$;
    else return $\perp$.

INTERFACE Add_2.hon(s_1, s_2):
If $\exists x_1, x_2 \in \mathbb{Z}_p$ so that
    $\mathbf{Label_2}.hon(x_1) = s_1$ and $\mathbf{Label_2}.hon(x_2) = s_2$
    then $x_3 \leftarrow x_1 + x_2 \bmod p$;
        return $\mathbf{Label_2}.hon(x_3)$;
    else return $\perp$.

INTERFACE Mult.hon(s_1, s_2):
If $\exists x_1, x_2 \in \mathbb{Z}_p$ so that
    $\mathbf{Label_1}.hon(x_1) = s_1$ and $\mathbf{Label_2}.hon(x_2) = s_2$
    then $x_3 \leftarrow x_1 \cdot x_2 \bmod p$;
        return $\mathbf{Label_T}.hon(x_3)$;
    else return $\perp$.
```

```
procedure GBM.adv:
INTERFACE Label_1.adv(x):
return $\mathbf{Label_1}.hon(x)$.

INTERFACE Label_2.adv(x):
return $\mathbf{Label_2}.hon(x)$.

INTERFACE Label_T.adv(x):
return $\mathbf{Label_T}.hon(x)$.

INTERFACE Add_1.adv(s_1, s_2):
return $\mathbf{Add_1}.hon(s_1, s_2)$.

INTERFACE Add_2.adv(s_1, s_2):
return $\mathbf{Add_2}.hon(s_1, s_2)$.

INTERFACE Mult.adv(s_1, s_2):
return $\mathbf{Mult}.hon(s_1, s_2)$.
```

Figure 5: Procedures implementing the functionality of the generic bilinear group model (GBM). The functionality is associated with $\mathbb{Z}_p$ and sets S_1, S_2, S_T.

```
procedure ICM.hon:
INTERFACE $\mathbf{E}.hon(k, x)$:
If $\mathbb{T}[k, x] = \perp$ then (re)sample $\mathbb{T}[k, x] \overset{\text{re}}{\twoheadleftarrow} \mathcal{R}$;
return $\mathbb{T}[x]$.

INTERFACE $\mathbf{E}^{-1}.hon(k, y)$:
If $\exists x$ so that $\mathbb{T}[k, x] = y$,
    return $x$;
else (re)sample $x \overset{\text{re}}{\twoheadleftarrow} \mathcal{R}$, $\mathbb{T}[k, x] = y$,
return $x$.

procedure ICM.adv:
INTERFACE $\mathbf{E}.adv(x)$:
return $\mathbf{E}.hon(x)$.

INTERFACE $\mathbf{E}^{-1}.adv(y)$:
return $\mathbf{E}^{-1}.hon(y)$.
```

Figure 6: Procedures implementing the functionality of the ideal cipher model (ICM). The functionality is associated with randomness space $\mathcal{R} = \{0, 1\}^r$ where the number $r \in \mathbb{N}$ is set as appropriate for a given context.

B Proof of Theorem 3.3

In this section, we complete the rest proof for the indifferentiable Schnorr signature. The following are the hybrids and we then show that any adjacent hybrids are statistically close.

B.1 Hybrid: $\mathcal{S}_2$

Simulator $\mathcal{S}_2$

The simulator $\mathcal{S}_2$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_2$ has the external oracle access to $\mathcal{F}_2$; the simulator $\mathcal{S}_2$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_2^{\mathbf{H}_{\text{sk}}}(SK)$: = $\mathcal{S}_1^{\mathbf{H}_{\text{sk}}}(SK)$

$\mathcal{S}_2^{\mathbf{Label}}(sk)$:= $\mathcal{S}_1^{\mathbf{Label}}(sk)$

$\mathcal{S}_2^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$:= $\mathcal{S}_1^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\mathcal{S}_2^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$:= $\mathcal{S}_1^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$

$\mathcal{S}_2^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$:= $\mathcal{S}_1^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$

$\mathcal{S}_2^{\mathbf{E}}(PK||M, V_1||V_2)$:= $\mathcal{S}_1^{\mathbf{E}}(PK||M, V_1||V_2)$

$\mathcal{S}_2^{\mathbf{E}^{-1}}(PK||M, V)$:
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $(V_1, V_2)$.
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $(V_1, V_2)$.
query the external $\mathcal{F}_2$ with ($\mathbf{Verify}$, $PK||M, V$), and obtain $\widehat{\phi}$;
if $\widehat{\phi} = 0$,  //for the invalid signature, respond with random string
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
if $\exists(SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big)$,
//Bad event: secret key collision, then responds with random string
    then $(V_1, V_2) \twoheadleftarrow \mathbb{Z}_p$ , $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
```

```
if $\exists(SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\rm sk}}$ or $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$ ,
    then $\widehat{sk} \leftarrow sk$  //identify the secret key ;
    else $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V), \mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
$\widehat{seed} \leftarrow \diamond$,
for each tuple $(PK^*, SK^*, M^*, R^*, seed^*) \in \mathbb{T}_{\mathbf{H}_{\rm seed}}$,  //identify the nonce
    query the external $\mathcal{F}_2$ with $(\mathbf{Sign}, PK^*, SK^*, M^*, R^*)$, and obtain $V^*$,
    if $PK^* = PK, M^* = M, V^* = V$,
        then $\widehat{seed} \leftarrow seed^*$;
if $\widehat{seed} = \diamond$,
    then $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V), \mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
$g_{\widehat{seed}} \leftarrow \mathcal{S}_2^{\mathbf{Label}}(\widehat{seed})$; $test \leftarrow \mathcal{S}_2^{\mathbf{H}_{\rm test}}(PK, M, g_{\widehat{seed}})$;
//compute the Schnorr signature $(V_1, V_2)$ using $\widehat{sk}$, $\widehat{seed}$ and test
$V_1 \leftarrow \widehat{seed} - \widehat{sk} \cdot test$, $V_2 \leftarrow test$,
$\mathbb{T}_{\mathbf{E}^{-1}} \leftarrow \mathbb{T}_{\mathbf{E}^{-1}} \cup \{(V, PK, M, V_1, V_2, seed)\}$;
return $(V_1, V_2)$.
```

Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ are identical except for responding to the $\mathbf{E}^{-1}$ queries, which are highlighted in the blue box. Essentially, there are two bad events that differ $\mathcal{S}_1$ and $\mathcal{S}_2$:

- Case 1: $\mathcal{F}_2.\textsc{Verify}(PK||M, V) = 0$;
- Case 2: Secret key collision.

For the first case, we immediately observe that V is not a valid signature. Let $(V_1, V_2) \leftarrow \mathcal{S}_1^{\mathbf{E}^{-1}}(PK||M, V)$ and $(V_1', V_2') \leftarrow \mathcal{S}_2^{\mathbf{E}^{-1}}(PK||M, V)$. By the description of $\mathcal{S}_2$, we note that V never appears via the adversarial interfaces, otherwise V would be stored in either $\mathbb{T}_{\mathbf{E}}$ or $\mathbb{T}_{\mathbf{E}^{-1}}$. Moreover, V can not be obtained by calling $\mathcal{F}_2.\textsc{Sign}(PK, SK, M, R)$ via the honest interface as V is an invalid signature. Thus, the differentiator view is independent of (V_1, V_2) except that

$$V_2 \neq \mathbf{H}_{\rm test}(PK, M, \mathbf{Label}(V_1 + sk\, V_2)).$$

Due to $\mathbf{H}_{\rm test}$ is a random oracle, it's apparent that the statistical distance $\Delta\Big((V_1, V_2), (V_1', V_2')\Big) \leq \frac{1}{p}$, which means the view on $\mathbf{E}^{-1}$ are close in either hybrid.

Moreover, we note that, for either $\mathcal{S}_1$ or $\mathcal{S}_2$, the response of the queries $\mathbf{H}_{\rm sk}, \mathbf{Label}, \mathbf{Add}, \mathbf{H}_{\rm seed}, \mathbf{H}_{\rm test}$ are identical. While for $\mathbf{E}$, we note that, after the query $\mathbf{E}^{-1}(PK||M, V)$ (denoted as Q below for ease), $\mathcal{S}_2$ implicitly sets $\mathbf{E}(PK||M, V_1'||V_2') = V$. Thus, as long as the differentiator never makes query $\mathbf{E}(PK||M, V_1'||V_2')$ before Q, and never makes a query $\mathbf{E}(PK||M, V_1||V_2)$ after Q, then the response of $\mathbf{E}$ queries are identical for either $\mathcal{S}_1$ or $\mathcal{S}_2$. And it's apparent that the bad event is bounded by $\frac{q}{p^2} + \frac{q(p-1)}{p^3}$.

For the second case, we argue that it never occurs except for negligible probability. In fact, $\mathbf{H}_{\rm sk}$ is a random oracle, and $\mathbf{GGM}$ is a generic group model, thus collision is trivially bounded by $\frac{q^2}{p}$.

Combing together, we have that $\Big|\Pr[\mathcal{G}_{\mathcal{S}_1}^{\mathcal{F}_1,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_2}^{\mathcal{F}_2,\mathcal{D}} = 1]\Big| \leq q(\frac{q}{p} + \frac{q}{p^2} + \frac{q(p-1)}{p^3}) + \frac{q^2}{p}$.

B.2 Hybrid: $\mathcal{S}_3$

Simulator $\mathcal{S}_3$

The simulator $\mathcal{S}_3$ will provide internal copies of the random oracles, $\mathbf{H}_{\rm sk}, \mathbf{H}_{\rm seed}, \mathbf{H}_{\rm test}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{$\textsc{Gen}$, $\textsc{Sign}$, $\textsc{Verify}$ }; the simulator $\mathcal{S}_3$ has the external oracle access to $\mathcal{F}_3$; the simulator $\mathcal{S}_3$ will provide the following interfaces for the external differentiator $\mathcal{D}$:
$\underline{\mathcal{S}_3^{\mathbf{H}_{\rm sk}}(SK)}$:= $\mathcal{S}_2^{\mathbf{H}_{\rm sk}}(SK)$

```
$\underline{\mathcal{S}_3^{\mathbf{Label}}(sk)}$:= $\mathcal{S}_2^{\mathbf{Label}}(sk)$

$\underline{\mathcal{S}_3^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$:
If $\nexists(sk_1, g_{sk_1}) \in \mathbb{T}_{\mathbf{Label}}$,
    then $sk_1 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \cup \{(\diamond, sk_1, g_{sk_1})\}$,
        $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk_1, g_{sk_1})\}$
If $\nexists(sk_2, g_{sk_2}) \in \mathbb{T}_{\mathbf{Label}}$,
    then $sk_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \cup \{(\diamond, sk_2, g_{sk_2})\}$,
        $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk_2, g_{sk_2})\}$
$sk \leftarrow sk_1 + sk_2 \bmod p$, return $\mathcal{S}_1^{\mathbf{Label}}(sk)$.
$\underline{\mathcal{S}_3^{\mathbf{H}_{\mathrm{seed}}}(PK, SK, M, R)}$: = $\mathcal{S}_2^{\mathbf{H}_{\mathrm{seed}}}(PK, SK, M, R)$

$\underline{\mathcal{S}_3^{\mathbf{H}_{\mathrm{test}}}(PK, M, \widehat{g})}$:= $\mathcal{S}_2^{\mathbf{H}_{\mathrm{test}}}(PK, M, \widehat{g})$

$\underline{\mathcal{S}_3^{\mathbf{E}}(PK||M, V_1||V_2)}$: = $\mathcal{S}_2^{\mathbf{E}}(PK||M, V_1||V_2)$

$\underline{\mathcal{S}_3^{\mathbf{E}^{-1}}(PK||M, V)}$:
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $(V_1, V_2)$.
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $(V_1, V_2)$.
query the external $\mathcal{F}_3$ with $(\mathbf{Verify}, PK||M, V)$, and obtain $\widehat{\phi}$;
if $\widehat{\phi} = 0$, //for the invalid signature, respond with random string
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
if $\exists(SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}\Big)$,
//Bad event: secret key collision, then responds with random string
    then $(V_1, V_2) \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
if $\exists(SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}$ or $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$ ,
    then $\widehat{sk} \leftarrow sk$ //identify the secret key ;
    else $\widehat{sk} \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \cup \{(\diamond, \widehat{sk}, PK)\}$; $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(\widehat{sk}, PK)\}$.
$\widehat{seed} \leftarrow \diamond$,
for each tuple $(PK^*, SK^*, M^*, R^*, seed^*) \in \mathbb{T}_{\mathbf{H}_{\mathrm{seed}}}$, //identify the nonce
    query the external $\mathcal{F}_3$ with $(\mathbf{Sign}, PK^*, SK^*, M^*, R^*)$, and obtain $V^*$,
    if $PK^* = PK, M^* = M, V^* = V$,
        then $\widehat{seed} \leftarrow seed^*$;
if $\widehat{seed} = \diamond$,
    then $(V_1, V_2) \leftarrow \mathbf{E}^{-1}(PK||M, V)$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$;
    return $(V_1, V_2)$.
$g_{\widehat{seed}} \leftarrow \mathcal{S}_3^{\mathbf{Label}}(\widehat{seed})$; $test \leftarrow \mathcal{S}_3^{\mathbf{H}_{\mathrm{test}}}(PK, M, g_{\widehat{seed}})$;
//compute the Schnorr signature $(V_1, V_2)$ using $\widehat{sk}$, $\widehat{seed}$ and $test$
$V_1 \leftarrow \widehat{seed} - \widehat{sk} \cdot test$, $V_2 \leftarrow test$,
$\mathbb{T}_{\mathbf{E}^{-1}} \leftarrow \mathbb{T}_{\mathbf{E}^{-1}} \cup \{(V, PK, M, V_1, V_2, seed)\}$;
return $(V_1, V_2)$.
```

Note that $\mathcal{S}_2$ and $\mathcal{S}_3$ are identical except for responding to the $\mathbf{Add}$ and $\mathbf{E}^{-1}$ queries, which are highlighted in the blue box. Essentially, there are two bad events that differ $\mathcal{S}_2$ and $\mathcal{S}_3$:

- Case 1: for query $\mathbf{Add}(g_{sk_1}, g_{sk_2})$ there is no $(sk_1, g_{sk_1}) \in \mathbb{T}_{\mathbf{Label}}$ or $(sk_2, g_{sk_2}) \in \mathbb{T}_{\mathbf{Label}}$,
- Case 2: for query $\mathbf{E}^{-1}(PK||M, V_1, V_2)$, there is no tuple $(SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}$ and $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$.

Note that, assume that the differentiator $\mathcal{D}$ proposes a group element, say g_{sk}, and we denote sk to be the value such that $\mathbf{Label}(sk) = g$, and $sk' \twoheadleftarrow \mathbb{Z}_p$. Next, we prove that, in $\mathcal{D}$'s view, the distance of sk and sk' are close. Essentially, if there is no tuple $(SK, sk, g_{sk}), (\diamond, sk, g_{sk}) \in \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}}$ or $(sk, g_{sk}) \in \mathbb{T}_{\mathbf{Label}}$ ($\mathcal{D}$ might know SK such that $\mathbf{Gen}(SK) = g_{sk}$), then the only way for the $\mathcal{D}$ to obtain to learn the $\mathbf{H}_{\mathrm{sk}}(SK)$ is via the interface $\mathbf{E}$. If $\mathcal{D}$ outputs (V_1, V_2) such that $seed = V_1 + \mathbf{H}_{\mathrm{sk}}(SK)\,V_2$, where $seed$ is known by $\mathcal{D}$, then $\mathcal{D}$ solves $\mathbf{H}_{\mathrm{sk}}(SK)$ directly. Otherwise, assume $seed \neq V_1 + \mathbf{H}_{\mathrm{sk}}(SK)\,V_2$, then $\mathcal{D}$ knows that $\mathbf{H}_{\mathrm{sk}}(SK) \neq \frac{seed - V_1}{V_2}$. The former case never occurs except for the negligible probability, bounded by $\frac{1}{p}$. The latter case, only rules out at most q candidates for $\mathbf{H}_{\mathrm{sk}}(SK)$. Thus, the statistical distance

$$\Delta(sk, sk') \le \frac{q}{p} + \frac{q}{p} + (p-q)\left(\frac{1}{p-q} - \frac{1}{p}\right) = \frac{3q}{p}.$$

Due to the negligible distance, it would be fine for $\mathcal{S}_3$ to implicitly set $\mathcal{S}_3^{\mathbf{H}_{\mathrm{sk}}}(SK) = sk'$, after the $\mathbf{Add}$ or $\mathbf{E}^{-1}$ query. Besides, the response for $\mathbf{Label}, \mathbf{H}_{\mathrm{seed}}, \mathbf{H}_{\mathrm{test}}$ are identical for either $\mathcal{S}_2$ or $\mathcal{S}_3$. While, for $\mathbf{E}$, it's apparent that if $\mathcal{D}$ does not makes a query $\mathbf{E}(g_{sk}||M, V_1, V_2)$ such that $seed = V_1 + sk'\,V_2$, then the responses for $\mathbf{E}$ are also identical, and sk' is uniformly sampled by $\mathcal{S}_3$, thus this bad event is bounded by $\frac{q}{p}$. Combing together, we have that

$$\left|\Pr[\mathcal{G}_{\mathcal{S}_2}^{\mathcal{F}_2,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_3}^{\mathcal{F}_3,\mathcal{D}} = 1]\right| \le q\left(\frac{(3q)}{p} + \frac{q}{p}\right) = \frac{4q^2}{p}.$$

B.3 Hybrid $\mathcal{S}_4$

Simulator $\mathcal{S}_4$

The simulator $\mathcal{S}_4$ will provide internal copies of the random oracles, $\mathbf{H}_{\mathrm{sk}}$, $\mathbf{H}_{\mathrm{seed}}$, $\mathbf{H}_{\mathrm{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_4$ has the external oracle access to $\mathcal{F}_4$; the simulator $\mathcal{S}_4$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_4^{\mathbf{H}_{\mathrm{sk}}}(SK)$: = $\mathcal{S}_3^{\mathbf{H}_{\mathrm{sk}}}(SK)$

$\mathcal{S}_4^{\mathbf{Label}}(sk)$:
if $\exists(sk, g_{sk}) \in \mathbb{T}_{\mathbf{Label}}$, then return $g_{sk}$;
$SK \twoheadleftarrow \mathcal{SK}$, query the $\mathcal{F}_4$ with ($\mathbf{Gen}$, $SK$), and obtain $\widehat{PK}$;
$\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, \widehat{PK})\}$; $\mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\mathrm{sk}}} \cup \{(SK, sk, \widehat{PK})\}$;
return $\widehat{PK}$.

$\mathcal{S}_4^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$: = $\mathcal{S}_3^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\mathcal{S}_4^{\mathbf{H}_{\mathrm{seed}}}(PK, SK, M, R)$: = $\mathcal{S}_3^{\mathbf{H}_{\mathrm{seed}}}(PK, SK, M, R)$

$\mathcal{S}_4^{\mathbf{H}_{\mathrm{test}}}(PK, M, \widehat{g})$:= $\mathcal{S}_3^{\mathbf{H}_{\mathrm{test}}}(PK, M, \widehat{g})$

$\mathcal{S}_4^{\mathbf{E}}(PK||M, V_1||V_2)$:= $\mathcal{S}_3^{\mathbf{E}}(PK||M, V_1||V_2)$

$\mathcal{S}_4^{\mathbf{E}^{-1}}(PK||M, V)$: = $\mathcal{S}_3^{\mathbf{E}^{-1}}(PK||M, V)$
```

Note that $\mathcal{S}_3$ and $\mathcal{S}_4$ are identical except for responding to the $\mathbf{Label}$ queries, which is highlighted in the blue box. Essentially, the only case that differ $\mathcal{S}_3$ and $\mathcal{S}_4$ is the following: for query $\mathbf{Label}(sk)$, there is no $(sk, g_{sk}) \in \mathbb{T}_{\mathbf{Label}}$. We denote that SK to be the secret key such that $\mathbf{H}_{\mathrm{sk}}(SK) = sk$[6] and $SK' \twoheadleftarrow \mathcal{SK}$.

[6] SK exists with overwhelming probability, $\ge 1 - e^{-2\lambda}$

We next prove that, in $\mathcal{D}$'s view, the distance of SK and SK' are close.

Essentially, the differentiator knows sk, it can generate valid (V_1, V_2) for $\mathbf{E}$ and tests the validity of the PK and learn information of the secret key SK. If $\mathcal{D}$ makes a query $PK \leftarrow \mathbf{Gen}(SK)$ such that $\mathbf{Verify}(PK||M, \mathbf{E}(PK||M, V_1||V_2)) = 1$, then it directly knows $PK = \mathbf{Label}(sk)$, and if not $\mathcal{D}$ rules out at most q candidates for SK. Thus, the statistical distance

$$\Delta(SK, SK') \leq \frac{q}{p} + \frac{q}{|\mathcal{SK}|} + (|\mathcal{SK}| - q)(\frac{1}{|\mathcal{SK}| - q} - \frac{1}{|\mathcal{SK}|}) \leq \frac{3q}{p}.$$

However, $\mathcal{S}_4$ implicitly set $\mathbf{H}_{\text{sk}}(\widehat{SK}) = sk$ such that $\mathbf{Gen}(SK') = \mathbf{Gen}(\widehat{SK})$, which in fact is not well distributed as sk is chosen by $\mathcal{D}$. Fortunately, we note that SK' is sampled by $\mathcal{S}_4$ and as long as $\mathcal{D}$ cannot make sure a query (bounded by $\frac{q}{p}$), the view on $\mathbf{H}_{\text{sk}}$ is perfectly proper. Moreover, if $\mathbf{Label}(sk)$ never appears (bounded by $\frac{q}{p}$), then the view on other interfaces are also good. Combing together, we have that $\left|\Pr[\mathcal{G}_{\mathcal{S}_3}^{\mathcal{F}_3,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_4}^{\mathcal{F}_4,\mathcal{D}} = 1]\right| \leq q(\frac{5q}{p})$.

B.4 Hybrid $\mathcal{S}_5$

Simulator $\mathcal{S}_5$

The simulator $\mathcal{S}_5$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group **GGM** = (**Label**, **Add**), and ideal cipher **ICM** =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_5$ has the external oracle access to $\mathcal{F}_5$; the simulator $\mathcal{S}_5$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_5^{\mathbf{H}_{\text{sk}}}(SK)}$:= $\mathcal{S}_4^{\mathbf{H}_{\text{sk}}}(SK)$

$\underline{\mathcal{S}_5^{\mathbf{Label}}(sk)}$:= $\mathcal{S}_4^{\mathbf{Label}}(sk)$

$\underline{\mathcal{S}_5^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$:= $\mathcal{S}_4^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\underline{\mathcal{S}_5^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)}$: = $\mathcal{S}_4^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$

$\underline{\mathcal{S}_5^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})}$: = $\mathcal{S}_4^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$

$\underline{\mathcal{S}_5^{\mathbf{E}}(PK||M, V_1||V_2)}$:
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $V$;
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $V$;
if $\exists(SK \neq SK')$ s.t. $\big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\big) \wedge \big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\big)$
    then goto Case 1; //Bad event: secret key collision
if $\nexists(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$,
    then goto Case 1; //Bad event: no knowledge of the corresponding secret key
    else $\widehat{SK} \leftarrow SK$; $\widehat{sk} \leftarrow sk$;
$seed \leftarrow V_1 + V_2 \cdot \widehat{sk} \bmod p$;
if $\nexists(seed, g_{seed}) \in \mathbb{T}_{\mathbf{Label}}$, then $g_{seed} \leftarrow \mathcal{S}_5^{\mathbf{Label}}(seed)$;
if $V_2 \neq \mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, g_{seed})$, //Bad event: $(V_1, V_2)$ fails the verification phase
    then go to Case 1;
if $\exists(PK, \widehat{SK}, M, R, seed) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$,
    then $\widehat{R} \leftarrow R$; go to Case 3;
    else go to Case 2. //Semi-bad event: $(V_1, V_2)$ passes verification with abused nonce

Case 1: //One of the bad events occurs, then responds with random string
        $V \twoheadleftarrow \mathbf{\Sigma}$; $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
```

```
Case 2:   //Semi-bad event occurs: then responds with a valid signature by sampling a nonce
          $\widehat{R} \leftarrow \mathcal{R}$; $\mathbb{T}_{\mathbf{H}_{\text{seed}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{seed}}} \cup \{PK, \widehat{SK}, M, \widehat{R}, seed\}$,
          query the external SIG with $(\mathbf{Sign}, PK, \widehat{SK}, M, \widehat{R})$, and obtain $V$;
          $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
Case 3:   query the external $\mathcal{F}_5$ with $(\mathbf{Sign}, PK, \widehat{SK}, M, \widehat{R})$, and obtain $V$;
          $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $V$.
$\mathcal{S}_5^{\mathbf{E}^{-1}}(PK||M, V)$:= $\mathcal{S}_4^{\mathbf{E}^{-1}}(PK||M, V)$
```

Note that $\mathcal{S}_4$ and $\mathcal{S}_5$ are identical except for responding to the $\mathbf{E}$ queries, which are highlighted in blue box. For Case 2 (yellow box), we denote that R to be the nonce such that $\mathbf{H}_{\text{seed}}(PK, SK, M, R) = seed$, and $R' \leftarrow \mathcal{R}$. Applying the same analysis in Section B.3, we see that the gap between $\mathcal{S}_4$ and $\mathcal{S}_5$ on Case 2 is bounded by $\frac{5q^2}{p}$.

Now it's rest to show the gap on Case 1 (red box). According to the description of the simulator, there are three bad events:

- Event 1: Secret key collision;
- Event 2: No knowledge of sk;
- Event 3: (V_1, V_2) is invalid.

Note that, the first one has been analyzed in Section B.1 and it occurs only within negligible probability. For the third one, if (V_1, V_2) is invalid, then $\mathbf{Verify}(PK||M, V) = 0$, and V is uniformly sampled which means V passes the verification phase with negligible probability ($\leq \frac{q}{p}$). For the second one, we note that the only chance that $\mathcal{D}$ can differ $\mathcal{S}_4$ and $\mathcal{S}_5$ is "outputs a valid $(V_1, V_2$ without knowing sk), which is trivially bounded by $\frac{q}{p}$. Combing together, we have that $\left|\Pr[\mathcal{G}_{\mathcal{S}_4}^{\mathcal{F}_4,\mathcal{D}} = 1] - \Pr[\mathcal{G}_{\mathcal{S}_5}^{\mathcal{F}_5,\mathcal{D}} = 1]\right| \leq \frac{7q^2}{p}$.

B.5 Hybrid $\mathcal{S}_6$

Simulator $\mathcal{S}_6$

The simulator $\mathcal{S}_6$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group **GGM** = (**Label**, **Add**), and ideal cipher **ICM** =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_6$ has the external oracle access to $\mathcal{F}_6$; the simulator $\mathcal{S}_6$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\mathcal{S}_6^{\mathbf{H}_{\text{sk}}}(SK)$:
if $\exists(SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$, then return $sk$;
query the external $\mathcal{F}_6$ with $(\mathbf{Gen}, SK)$, and obtain $\widehat{PK}$;
if $\exists(\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$ s.t. $PK = \widehat{PK}$, then return $sk$;
$sk \twoheadleftarrow \mathbb{Z}_p$; $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(sk, \widehat{PK})\}$; $\mathbb{T}_{\mathbf{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{sk}}} \cup \{(SK, sk, \widehat{PK})\}$;
return $sk$.

$\mathcal{S}_6^{\mathbf{Label}}(sk)$: = $\mathcal{S}_5^{\mathbf{Label}}(sk)$

$\mathcal{S}_6^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$: = $\mathcal{S}_5^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\mathcal{S}_6^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$:= $\mathcal{S}_5^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$

$\mathcal{S}_5^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$: = $\mathcal{S}_6^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$

$\mathcal{S}_6^{\mathbf{E}}(PK||M, V_1||V_2)$:= $\mathcal{S}_5^{\mathbf{E}}(PK||M, V_1||V_2)$
```

```
$\underline{\mathcal{S}_6^{\mathbf{E}^{-1}}(PK||M, V)}$:= $\mathcal{S}_5^{\mathbf{E}^{-1}}(PK||M, V)$
```

Applying exactly the same analysis in Section B.2, we trivially have that

$$\big|\Pr[\mathcal{G}_{\mathcal{S}_5}^{\mathcal{F}_5,\mathcal{D}}=1]-\Pr[\mathcal{G}_{\mathcal{S}_6}^{\mathcal{F}_6,\mathcal{D}}=1]\big| \leq \big|\Pr[\mathcal{G}_{\mathcal{S}_2}^{\mathcal{F}_2,\mathcal{D}}=1]-\Pr[\mathcal{G}_{\mathcal{S}_3}^{\mathcal{F}_3,\mathcal{D}}=1]\big|$$

B.6 Hybrid $\mathcal{S}_7$

Simulator $\mathcal{S}_7$

The simulator $\mathcal{S}_7$ will provide internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group **GGM** = (**Label**, **Add**), and ideal cipher **ICM** =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_7$ has the external oracle access to $\mathcal{F}_7$; the simulator $\mathcal{S}_7$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

```
$\underline{\mathcal{S}_7^{\mathbf{H}_{\text{sk}}}(SK)}$: = $\mathcal{S}_6^{\mathbf{H}_{\text{sk}}}(SK)$

$\underline{\mathcal{S}_7^{\mathbf{Label}}(sk)}$: = $\mathcal{S}_6^{\mathbf{Label}}(sk)$

$\underline{\mathcal{S}_7^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$: = $\mathcal{S}_6^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\underline{\mathcal{S}_7^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R):}$
$\widehat{PK} \leftarrow \mathcal{S}_7^{\mathbf{H}_{\text{sk}}}(SK)$;
if $\exists(PK, SK, M, R, seed) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$,
    then return $seed$;
query the external $\mathcal{F}_7$ with ($\mathbf{Sign}$, $PK, SK, M, R$), and obtain $\widehat{V}$,
if $\exists(\widehat{V}, PK, M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$,
    then return $seed$;
$seed \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\text{seed}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{seed}}} \cup \{(PK, SK, M, R, seed)\}$,
return $seed$.

$\underline{\mathcal{S}_7^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})}$:
if $\exists(PK, M, \widehat{g}, test) \in \mathbb{T}_{\mathbf{H}_{\text{test}}}$, then return $test$;
$test \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\text{test}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{test}}} \cup \{(PK, M, \widehat{g}, test)\}$,
return $test$.

$\underline{\mathcal{S}_7^{\mathbf{E}}(PK||M, V_1||V_2)}$:= $\mathcal{S}_6^{\mathbf{E}}(PK||M, V_1||V_2)$

$\underline{\mathcal{S}_7^{\mathbf{E}^{-1}}(PK||M, V)}$:
if $\exists(V, PK||M, V_1, V_2) \in \mathbb{T}_{\mathbf{E}}$, then return $(V_1, V_2)$.
if $\exists(V, PK||M, V_1, V_2, seed) \in \mathbb{T}_{\mathbf{E}^{-1}}$, then return $(V_1, V_2)$.
query the external $\mathbf{SIG}$ with ($\mathbf{Verify}$, $PK||M, V$), and obtain $\widehat{\phi}$;
if $\widehat{\phi} = 0$, //for the invalid signature, respond with random strings
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $(V_1, V_2)$.
if $\exists(SK \neq SK')$ s.t. $\Big((SK, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big) \wedge \Big((SK', sk', PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}\Big)$,
//Bad event: secret key collision, then responds with random strings
    then $V_1, V_2 \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{E}} \leftarrow \mathbb{T}_{\mathbf{E}} \cup \{(V, PK, M, V_1, V_2)\}$; return $(V_1, V_2)$.
if $\exists(SK, sk, PK), (\diamond, sk, PK) \in \mathbb{T}_{\mathbf{H}_{\text{sk}}}$ or $(sk, PK) \in \mathbb{T}_{\mathbf{Label}}$ ,
    then $\widehat{sk} \leftarrow sk$ //identify the secret key ;
    else $\widehat{sk} \twoheadleftarrow \mathbb{Z}_p$, $\mathbb{T}_{\mathbf{H}_{\text{sk}}} \leftarrow \mathbb{T}_{\mathbf{H}_{\text{sk}}} \cup \{(\diamond, \widehat{sk}, PK)\}$; $\mathbb{T}_{\mathbf{Label}} \leftarrow \mathbb{T}_{\mathbf{Label}} \cup \{(\widehat{sk}, PK)\}$
$\widehat{seed} \twoheadleftarrow \mathbb{Z}_p$,
for each tuple $(PK^*, SK^*, M^*, R^*, seed^*) \in \mathbb{T}_{\mathbf{H}_{\text{seed}}}$, //identify the nonce
    query the external $\mathbf{SIG}$ with ($\mathbf{Sign}$, $PK^*, SK^*, M^*, R^*$), and obtain $V^*$,
    if $PK^* = PK, M^* = M, V^* = V$,
        then $\widehat{seed} \leftarrow seed^*$;
```

$g_{\widehat{seed}} \leftarrow \mathcal{S}^{\mathbf{Label}}(\widehat{seed})$; $test \leftarrow \mathcal{S}^{\mathbf{H}_{\text{test}}}(PK, M, g_{\widehat{seed}})$;

//compute the Schnorr signature (V_1, V_2) *using* $\widehat{sk}$, $\widehat{seed}$ *and test*

$V_1 \leftarrow \widehat{seed} - \widehat{sk} \cdot test$, $V_2 \leftarrow test$,

$\mathbb{T}_{\mathbf{E}^{-1}} \leftarrow \mathbb{T}_{\mathbf{E}^{-1}} \cup \{(V, PK, M, V_1, V_2, seed)\}$;

return (V_1, V_2).

Note that the difference between $\mathcal{S}_6$ and $\mathcal{S}_7$ are the queries to $\mathbf{H}_{\text{seed}}$ and $\mathbf{H}_{\text{test}}$. Concretely, $\mathcal{S}_6$ responds to those queries by calling $\mathbf{H}_{\text{seed}}$ and $\mathbf{H}_{\text{test}}$, while $\mathcal{S}_7$ replaces the response by sampling random strings. Observe that $\mathbf{H}_{\text{seed}}$ and $\mathbf{H}_{\text{test}}$ are random oracles, thus the view of $\mathcal{D}$ on $\mathcal{S}_6$ is identical to the one on $\mathcal{S}_7$, referring to

$$\Pr[\mathcal{G}_{\mathcal{S}_6}^{\mathcal{F}_6,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_7}^{\mathcal{F}_7,\mathcal{D}} = 1].$$

B.7 Hybrid $\mathcal{S}_8$

Simulator $\mathcal{S}_8$

The simulator $\mathcal{S}_8$ will keep nothing of the internal copies of the random oracles, $\mathbf{H}_{\text{sk}}$, $\mathbf{H}_{\text{seed}}$, $\mathbf{H}_{\text{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$), and internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_8$ has the external oracle access to $\mathcal{F}_8$, where F_8 will provide the internal copy of Π =Π.{Gen, Sign, Verify }; the simulator $\mathcal{S}_8$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\underline{\mathcal{S}_8^{\mathbf{H}_{\text{sk}}}(SK)}$: = $\mathcal{S}_7^{\mathbf{H}_{\text{sk}}}(SK)$

$\underline{\mathcal{S}_8^{\mathbf{Label}}(sk)}$: = $\mathcal{S}_7^{\mathbf{Label}}(sk)$

$\underline{\mathcal{S}_8^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$: = $\mathcal{S}_7^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\underline{\mathcal{S}_8^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)}$:= $\mathcal{S}_7^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$

$\underline{\mathcal{S}_8^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})}$: =$\mathcal{S}_7^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$

$\underline{\mathcal{S}_7^{\mathbf{E}}(PK||M, V_1||V_2)}$:= $\mathcal{S}_6^{\mathbf{E}}(PK||M, V_1||V_2)$

$\underline{\mathcal{S}_8^{\mathbf{E}^{-1}}(PK||M, V)}$: = $\mathcal{S}_7^{\mathbf{E}^{-1}}(PK||M, V)$

Note that, when answering to the queries, $\mathcal{S}_7$ never calls the random oracles $\mathbf{H}_{\text{sk}}, \mathbf{H}_{\text{seed}}, \mathbf{H}_{\text{test}}$, generic group $\mathbf{GGM}$ = ($\mathbf{Label}$, $\mathbf{Add}$), and ideal cipher $\mathbf{ICM}$ =($\mathbf{E}$, $\mathbf{E}^{-1}$). Therefore, it's perfectly fine to remove the internal copies from $\mathcal{S}_7$, which means $\Pr[\mathcal{G}_{\mathcal{S}_7}^{\mathcal{F}_7,\mathcal{D}} = 1] = \Pr[\mathcal{G}_{\mathcal{S}_8}^{\mathcal{F}_8,\mathcal{D}} = 1]$.

B.8 Hybrid $\mathcal{S}_9$

Simulator $\mathcal{S}_9$

The simulator $\mathcal{S}_9$ keeps no internal copies; the simulator $\mathcal{S}_9$ has the external oracle access to $\mathbf{SIG}$ = ($\mathbf{Gen}$, $\mathbf{Sign}$, $\mathbf{Verify}$), the simulator $\mathcal{S}_9$ will provide the following interfaces for the external differentiator $\mathcal{D}$:

$\underline{\mathcal{S}_9^{\mathbf{H}_{\text{sk}}}(SK)}$: = $\mathcal{S}_8^{\mathbf{H}_{\text{sk}}}(SK)$

$\underline{\mathcal{S}_9^{\mathbf{Label}}(sk)}$: = $\mathcal{S}_8^{\mathbf{Label}}(sk)$

$\underline{\mathcal{S}_9^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})}$: = $\mathcal{S}_8^{\mathbf{Add}}(g_{sk_1}, g_{sk_2})$

$\underline{\mathcal{S}_9^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)}$:= $\mathcal{S}_8^{\mathbf{H}_{\text{seed}}}(PK, SK, M, R)$

$\underline{\mathcal{S}_9^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})}$: =$\mathcal{S}_8^{\mathbf{H}_{\text{test}}}(PK, M, \widehat{g})$

$\underline{\mathcal{S}_9^{\mathbf{E}}(PK||M, V_1||V_2)}$:= $\mathcal{S}_8^{\mathbf{E}}(PK||M, V_1||V_2)$

$\underline{\mathcal{S}_9^{\mathbf{E}^{-1}}(PK||M, V)}$: = $\mathcal{S}_8^{\mathbf{E}^{-1}}(PK||M, V)$

We immediately observe that the description of $\mathcal{S}_9$ is identical to $\mathcal{S}$ and it's apparent that $\Pr[\mathcal{G}_{\mathcal{S}_9}^{\mathcal{F}_9,\mathcal{D}}] = \Pr[\mathbf{Ideal}_{\mathcal{S}}^{\mathbf{SIG},\mathcal{D}}]$.

Hence, it suffices to show that $\mathcal{S}_8$ and $\mathcal{S}_9$ are close. In fact, in Game 8, the external oracle is $\Pi = (\Pi.\text{Gen}, \Pi.\text{Sign}, \Pi.\text{Verify})$, and in Game 9, the oracle is ideal signature $\mathbf{SIG} = (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$. According to the definition of ideal signature, it's trivial that if for any value $sk \in [p]$, there exists $SK \in \mathcal{SK}$ such that $\mathbf{H}_{\text{sk}}(SK) = sk$ and for any pair (PK, SK, M) and any value $r \in [p]$, there exists $R \in \mathcal{R}$ such that $\mathbf{H}_{\text{seed}}(PK, SK, M, R) = r$, then the distribution of Π and $\mathbf{SIG}$ are identical. And this bad event is bounded by

$$2p \cdot (1 - \frac{1}{p})^{p \cdot 2^\lambda} \leq 2p \cdot e^{-2^\lambda}$$

where e is the natural logarithm.

Combing all together, we complete the entire proof.