Designated Verifier/Prover and Preprocessing NIZKs from Diffie-Hellman Assumptions

Shuichi Katsumata[1,3], Ryo Nishimaki[2], Shota Yamada[3], Takashi Yamakawa[2]

[1]The University of Tokyo, Tokyo ,Japan
shuichi_katsumata@it.k.u-tokyo.ac.jp
[2]NTT Secure Platform Laboratories, Tokyo, Japan
{ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp
[3]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
yamada-shota@aist.go.jp

July 9, 2019

Abstract

In a non-interactive zero-knowledge (NIZK) proof, a prover can non-interactively convince a verifier of a statement without revealing any additional information. Thus far, numerous constructions of NIZKs have been provided in the common reference string (CRS) model (CRS-NIZK) from various assumptions, however, it still remains a long standing open problem to construct them from tools such as pairing-free groups or lattices. Recently, Kim and Wu (CRYPTO'18) made great progress regarding this problem and constructed the first lattice-based NIZK in a relaxed model called NIZKs in the preprocessing model (PP-NIZKs). In this model, there is a trusted statement-independent preprocessing phase where secret information are generated for the prover and verifier. Depending on whether those secret information can be made public, PP-NIZK captures CRS-NIZK, designated-verifier NIZK (DV-NIZK), and designated-prover NIZK (DP-NIZK) as special cases. It was left as an open problem by Kim and Wu whether we can construct such NIZKs from weak paring-free group assumptions such as DDH. As a further matter, all constructions of NIZKs from Diffie-Hellman (DH) type assumptions (regardless of whether it is over a paring-free or paring group) require the proof size to have a multiplicative-overhead $|C| \cdot \mathsf{poly}(\kappa)$, where $|C|$ is the size of the circuit that computes the **NP** relation.

In this work, we make progress of constructing (DV, DP, PP)-NIZKs with varying flavors from DH-type assumptions. Our results are summarized as follows:

- DV-NIZKs for **NP** from the CDH assumption over pairing-free groups. This is the first construction of such NIZKs on pairing-free groups and resolves the open problem posed by Kim and Wu (CRYPTO'18).
- DP-NIZKs for **NP** with short proof size from a DH-type assumption over pairing groups. Here, the proof size has an additive-overhead $|C| + \mathsf{poly}(\kappa)$ rather than an multiplicative-overhead $|C| \cdot \mathsf{poly}(\kappa)$. This is the first construction of such NIZKs (including CRS-NIZKs) that does not rely on the LWE assumption, fully-homomorphic encryption, indistinguishability obfuscation, or non-falsifiable assumptions.
- PP-NIZK for **NP** with short proof size from the DDH assumption over pairing-free groups. This is the first PP-NIZK that achieves a short proof size from a weak and static DH-type assumption such as DDH. Similarly to the above DP-NIZK, the proof size is $|C| + \mathsf{poly}(\kappa)$. This too serves as a solution to the open problem posed by Kim and Wu (CRYPTO'18).

Along the way, we construct two new homomorphic authentication (HomAuth) schemes which may be of independent interest.

1 Introduction

1.1 Background

Zero-knowledge (ZK) proof system [GMR89] is an interactive protocol where a prover convinces the validity of a statement to a verifier without providing any additional knowledge. A non-interactive zero-knowledge (NIZK) proof (or argument[1]) [BFM88] is a ZK proof (or argument) where a prover can generate a proof to the validity of a statement *without* interacting with a verifier. Due to the absence of interaction, NIZKs have found tremendous number of applications in cryptography including (but not limited to) chosen-ciphertext secure public key encryption [NY90, DDN00, Sah99], group/ring signatures [Cv91, BMW03, RST01], anonymous credentials [Cha85, Dam90], and multi-party computations (MPC) [GMW87]. Furthermore, aside from its practical interests, due to its theoretically appealing nature, studying the types of assumptions which imply NIZKs has also been an active research area for NIZKs [FLS99, GOS12, BPW16, RSS18]. Below, we briefly review the current state of affairs concerning NIZKs.

NIZKs in the CRS model. It is well known that NIZKs for non-trivial languages do not exist in the plain model where there is no trusted setup [GO94]. Therefore NIZKs for all of **NP** are constructed either in the common reference string (CRS) model [FLS99] or the random oracle model [FS87, PS00]. In the former type of NIZK, the prover and the verifier have access to a CRS generated by a trusted third party (hereafter referred to as CRS-NIZK). Thus far, known constructions of CRS-NIZK for **NP** are based on (doubly-enhanced) trapdoor permutation [FLS99, BY96, Gol04], pairing [GOS12, GS12], or indistinguishability obfuscation [SW14, BP15, BPW16]. Constructing CRS-NIZKs based on other assumptions such as pairing-free groups and lattices remains to be a long standing open problem.

NIZKs in the designated verifier/prover model. As an alternative line of research, NIZKs in a relaxed model have been considered: *designated verifier* NIZKs (DV-NIZKs) and *designated prover* NIZKs (DP-NIZKs). Both notions of NIZKs retain most of the useful security properties of NIZKs with some relaxation. In DV-NIZKs, anybody can generate a proof, but the proof can only be verified by a designated party in possession of a *verification key*. On the other hand, in DP-NIZKs only a designated party in possession of a *proving key* can generate a proof, but the proof can be verified by anybody. Although the two types of NIZKs are relaxation of CRS-NIZKs, they showed to be no easier to construct. There have been a long line of work concerning DV-NIZKs [PsV06, DFN06, VV09, CG15, Lip17, CC18, CDI+18], however, many of these schemes do not satisfy soundness against multiple theorems, which in brief means that soundness does not hold against a cheating prover given unbounded access to a verification oracle (See Section 1.4 for more details). Moreover, DV-NIZKs satisfying soundness against multiple theorems [CC18, CDI+18] are built on tools which are already known to imply CRS-NIZKs. Similarly to DV-NIZKs, it was not until recently that Kim and Wu [KW18a] in a breakthrough result showed how to construct DP-NIZKs supporting **NP** languages from lattices; this is the first NIZKs for all of **NP** in any model that is based on lattice assumptions. They showed a generic construction of DP-NIZKs from homomorphic signatures (HomSig) and instantiated it with the lattice-based HomSig of [GVW15b]. However, despite these recent developments, basing the construction of DV-NIZKs or DP-NIZKs for all of **NP** on pairing-free groups still remains unsolved, and Kim and Wu [KW18a] have stated it as an open problem to construct such NIZKs from the decisional Diffie-Hellman (DDH) assumption.

First Contribution. One of our main contributions is solving this open problem and constructing the first DV-NIZKs from the computational Diffie-Hellman (CDH) assumption over *pairing-free groups*. As our scheme is DV-NIZKs and not DP-NIZKs, our techniques depart from [KW18a] and follows more closely to the classical techniques of [FLS99]. More details will be provided in Section 1.2.

NIZKs with short proof size. An equally important topic for NIZKs is constructing NIZKs with short proof size. Our construction above solves the open problem of constructing DV or DP-NIZKs from pairing-free groups, however, the size of proof is rather large. Namely, it is of size $\mathsf{poly}(\kappa, |C|)$, where κ is the security parameter and $|C|$ is the size of circuit computing the **NP** relation $\mathcal{R}$. In particular, the proof size incurs at least a *multiplicative*-overhead of $O(|C|\kappa)$. As far as we know, the only (CRS, DV, DP)-NIZKs for **NP** in the standard model with a short proof size, i.e., a proof with *additive*-overhead $O(|C|) + \mathsf{poly}(\kappa)$ rather than $O(|C|) \cdot \mathsf{poly}(\kappa)$, either requires a knowledge assumption [Gro10], (fully-)homomorphic encryption (FHE) [GGI+15], indistinguishability obfuscation (iO) [SW14], or HomSig

[1] NIZK arguments are a relaxed notion of NIZK proofs where soundness only holds against computationally bounded adversaries. Throughout the introduction, we simply refer to them as NIZKs.

with additional compactness properties [KW18a].[2] Notably, we do not know how to construct (CRS, DV, DP)-NIZKs with short proof size from standard assumptions from pairing-free groups. In fact, this is the case even if we were to consider pairing groups [CHK07, GOS12, Abu13] as none of the aforementioned heavy machineries are implied from such groups. In other words, it is not known whether DH-type assumptions can be used to construct DV or DP-NIZKs with short proof size.

Second Contribution. Our second contribution is constructing a DP-NIZK *with short proof size* from a DH-type assumption over pairing groups by proposing a compact HomSig scheme from a new non-static DH-type assumption (proven to hold in the generic group model) and following the general conversion from HomSig to DP-NIZK by Kim and Wu [KW18a]. More details will be provided in Section 1.2.

Our second scheme achieves the first DP-NIZK with short proof size from any DH-type assumptions, however, one caveat is that the assumption is *non-static* and rather strong, and furthermore requires *pairing groups*. Therefore, desirably we would like to construct any type of NIZKs with short proof size from weaker and *static* assumptions such as the DDH assumption while only requiring *pairing-free groups*. To this end, we consider a further relaxation of NIZKs in the *preprocessing model* (hereafter referred to as PP-NIZK). In this model, there is a trusted preprocessing setup that generates a verification *and* proving key, where only those with the proving (resp. verification) key can generate (resp. verify) proofs. Analogously to the history of DV and DP-NIZKs, even with this added relaxation, PP-NIZKs turned out to be a rather difficult primitive to construct. There have been several works concerning PP-NIZKs [DMP90, KMO90, LS91, Dam93, CD04, IKOS09], however, all of them were only *bounded-theorem* in the sense that either the soundness or zero-knowledge property hold in a bounded manner (See Section 1.4 for more details). The problem of constructing *unbounded-theorem* PP-NIZKs, which meets the standard criteria of NIZK, was only recently resolved in the aforementioned paper [KW18a], where Kim and Wu showed a generic construction of PP-NIZKs using homomorphic MACs (HomMAC). In particular, depending on whether the signature can be verified publicly (HomSig) or not (HomMAC), their generic construction leads to a DP-NIZK or a PP-NIZK, respectively. In fact, it was observed in [KW18a] that using the compact HomMAC proposed by Catalano and Fiore [CF18] based on the non-static ℓ-computational DH inversion (ℓ-CDHI) assumption [BB04, CHL05], we can construct PP-NIZKs from a non-static DH-type assumption over pairing-free groups. However, they left it as an open problem to construct HomMAC that suffices for PP-NIZKs (with short proof size) from a weaker static assumption such as DDH.

Final Contribution. Our final contribution is constructing a PP-NIZK *with short proof size* from the DDH assumption over *pairing-free groups*. We first construct a non-compact HomMAC from the DDH assumption and exploit extra structures in our HomMAC to achieve short proof size when converting it into a PP-NIZK. More details will be provided in Section 1.2.

Motivation for studying different types of NIZKs. Although (DV, DP, PP)-NIZKs may be more restricted compared to CRS-NIZKs, they can be useful nonetheless. For example, applications of CRS-NIZKs including group signatures [Cv91, BMW03], anonymous credentials [Cha85, Dam90], electronic cash [CFN90], anonymous authentication [TFS04] may lead to a designated verifier or prover variant by using DV or DP-NIZKs. In some natural scenarios where we do not require public verifiability or require everybody to be able to construct proofs, these alternatives may suffice. Furthermore, as stated in [KW18a], PP-NIZKs can be used instead of CRS-NIZKs to boost semi-honest security to malicious security [GMW87]. Finally, we believe studying different types of NIZKs and understanding which assumptions imply them will provide us with new insights on realizing the long standing open problem of constructing CRS-NIZKs from pairing-free groups or lattices.

1.2 Our Results in Detail

As briefly mentioned above, we give new constructions of DV-NIZK, DP-NIZK, and PP-NIZK with different flavors from DH-type assumptions. Our first and third schemes are instantiated on a pairing-free group, and the second scheme requires a pairing group.

1. We construct DV-NIZKs for **NP** from the CDH assumption over pairing-free groups that resists the verifier rejection attack. This is the first construction of such (DV, DP)-NIZK on pairing-free groups and resolves the open problem posed by Kim and Wu [KW18a].

[2] In fact, as we show in Table 1, all of these approaches lead to a much more succinct proof size of $|w| + \mathsf{poly}(\kappa)$, where w is the witness.

2. We construct DP-NIZKs for **NP** with short proof size from a newly defined non-static (n, m)-computational DH exponent and ratio (CDHER) assumption (proven in the generic group model) over pairing groups. This is the first NIZK in the standard model to achieve a short proof size without assuming the LWE assumption, fully-homomorphic encryption, indistinguishability obfuscation, or non-falsifiable assumptions. The proof size has an additive-overhead $|C| + \mathsf{poly}(\kappa)$ rather than a multiplicative-overhead $|C| \cdot \mathsf{poly}(\kappa)$ where $|C|$ is the size of the circuit that computes the **NP** relation (See Table 1). Moreover, if we make a slight relaxation in the assumption that the **NP** relation is expressed by a "leveled circuit" [BGI16], then the proof size can be made as short as $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$ where $|w|$ denotes the witness size. This is the first NIZK (including PP-NIZKs) that achieves *sublinear* proof size in $|C|$. We note that by applying the same technique to the ℓ-CDHI-based construction of PP-NIZK stated in Kim and Wu [KW18a], we can make their proof size sublinear as well, as long as the **NP** relation can be expressed by a leveled circuit.

3. We construct PP-NIZKs for **NP** with short proof size from the DDH assumption over pairing-free groups that are multi-theorem. This is the first PP-NIZK that achieves a short proof size from a weak and static DH-type assumption such as DDH. (In fact, this construction also serves as a solution to the open problem posed by Kim and Wu [KW18a].) Similarly to the above DP-NIZK, the proof size is $|C| + \mathsf{poly}(\kappa)$. Moreover, going through the same technique with additional observations, in case the **NP** relation can be expressed by a leveled circuit, we are able to make the proof size sublinear $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$.

Perhaps of an independent interest, along the way to achieve our second result, we propose an HomSig scheme that simultaneously achieves compactness, context-hiding, and online-offline efficiency under the (n, m)-CDHER assumption. This is the first construction of such HomSig schemes on pairing groups.

The comparison table among existing and our NIZK is given in Table 1. We note that we omit schemes that do not support all of **NP**, do not resist the verifier rejection attack, or do not achieve unbounded-theorem soundness or zero-knowledge from the table.

Table 1: Comparison of NIZKs for **NP**.

Reference	Soundness	ZK	Proof size	Model	Assumption	Misc
[FLS99]	stat.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	CRS	trapdoor permutation‡	
[Gro10]	stat.	comp.	$\|C\| \cdot k_{\mathsf{tpm}} \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	CRS	trapdoor permutation‡	
[Gro10]	stat.	comp.	$\|C\| \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	CRS	Naccache-Stern PKE	
[GOS12]	perf.	comp.	$O(\|C\|\kappa)$	CRS	DLIN/SD	
[GOS12]	comp.	perf.	$O(\|C\|\kappa)$	CRS	DLIN/SD	
[CHK07, DN07, Abu13]	stat.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	CRS	CDH	
[Gro10]	comp.	perf.	$O(\kappa)$	CRS	q-PKE and q-CPDH	knowledge assumption
[GGI+15]	stat.	comp.	$\|w\| + \mathsf{poly}(\kappa)$	CRS	FHE and CRS-NIZK	
[SW14]	comp.	perf.	$O(\kappa)$	CRS	iO and OWF	
[KW18a]	stat.*	comp.	$\|w\| + \mathsf{poly}(\kappa, d)$	DP	LWE	
[CF18]+[KW18a]	comp.	comp.	$\|C\| + \mathsf{poly}(\kappa)$	PP	ℓ-CDHI	pairing-free
Section 3	stat.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	DV	CDH	pairing-free
Section 4	comp.	comp.	$\|C\| + \mathsf{poly}(\kappa)$	DP	(n, m)-CDHER	
Section 4+Appendix C.4†	comp.	comp.	$\|w\| + \|C\|/\log(\kappa) + \mathsf{poly}(\kappa)$	DP	(n, m)-CDHER	
Section 5	stat.	comp.	$\|C\| + \mathsf{poly}(\kappa)$	PP	DDH	pairing-free
Section 5+Appendix C.4†	stat.	comp.	$\|w\| + \|C\|/\log(\kappa) + \mathsf{poly}(\kappa)$	PP	DDH	pairing-free
[CF18]+Appendix C.4†	comp.	comp.	$\|w\| + \|C\|/\log(\kappa) + \mathsf{poly}(\kappa)$	PP	ℓ-CDHI	pairing-free

In column "Soundness" (resp. "ZK"), perf., stat., and comp. means perfect, statistical, and computational soundness (resp. zero-knowledge), respectively. In column "Proof size", κ is the security parameter, $|w|$ is the witness-size, $|C|$ and d are the size and depth of circuit computing the **NP** relation, and k_{tpm} is the length of the domain of the trapdoor permutation. In column "Assumption", DLIN stands for the decisional linear assumption, SD stands fo the subgroup decision assumption, q-PKE stands for the q-power knowledge of exponent assumption, and q-CPDH stands fo the q-computational power Diffie-Hellman assumption.

* Though their primary construction only has computational soundness, they sketched a variant that achieves statistical soundness in the latest version [KW18b, Remark 4.10]

† Applicable only when C is a leveled circuit.

‡ If the domain of the permutation is not $\{0,1\}^n$, we further assume they are doubly enhanced [Gol04].

1.3 Technical Overview

We rely on mainly two approaches to achieve our results. The first approach is an extension of the construction of CRS-NIZKs from trapdoor permutations by Feige, Lapidot, and Shamir [FLS99] (we call it the FLS construction) to the DV setting. The second approach is constructing (DP, PP)-NIZKs using the Kim-Wu conversion [KW18a] from homomorphic authenticators (HomAuth), where HomAuth are shorthand for HomSig and HomMAC. Specifically, we provide new instantiations of context-hiding HomAuth schemes. Our first result is obtained by the first approach, and the second and third results are obtained by the second approach. In the following, we explain these approaches.

Part 1: DV-NIZK from CDH via FLS paradigm. Our DV-NIZK is based on the Feige-Lapidot-Shamir (FLS) paradigm [FLS99], which enables to construct CRS-NIZKs based on trapdoor permutations (TDP). However, we can not directly use the FLS paradigm since we currently do not know how to achieve TDPs from the CDH assumption. In this study, we present a variant of the FLS construction in the DV setting that can be instantiated by the CDH assumption over paring-free groups.

Our starting point is the CRS-NIZK based on the CDH assumption *over pairing groups* [CHK07, DN07, Abu13]. The idea is to use a function f_ι defined as follows instead of a TDP for the FLS construction: $f_\iota(X, Z) := X$ if (g, X, Y, Z) is a DH tuple and otherwise $\perp$, where $\iota := (g, Y = g^\tau)$. Though f_ι is not a TDP, it is a trapdoor function (TDF) with a structure that is sufficient for implementing the FLS construction. Below, we take a closer look at the construction.

NIZK in the Hidden Bits Model. Before explaining the construction, we recall the notion of NIZK proof systems in the hidden bits model (hereafter referred to as HBM-NIZK) [FLS99]. In [FLS99], HBM-NIZKs is used as a building block for the final CRS-NIZK. In HBM-NIZK, a prover is provided with a randomly generated string $\rho \xleftarrow{\$} \{0,1\}^\ell$ (referred to as a *hidden random string*) independently from the statement x and witness w for the **NP** language $\mathcal{L}$. Then it generates a proof π_{hbm} along with an index set I indicating the positions in the hidden random string. A verifier given a sub-string $\rho_{|I}$ of the hidden random string ρ on positions corresponding to the index set I along with the statement x and a proof π_{hbm}, either accepts or rejects. Soundness requires that no adversary can generate a valid proof π_{hbm} with an index set I if $x \notin \mathcal{L}$, and the zero-knowledge property requires that a proof provides no additional knowledge to the verifier beyond that $x \in \mathcal{L}$ *if all bits of* ρ *on positions corresponding to* $[\ell] \setminus I$ *are hidden* to the verifier. Feige et al. proved that HBM-NIZKs for all of **NP** exist unconditionally.

CRS-NIZK from CDH with pairings We now describe the CRS-NIZK based on the CDH assumption over pairing groups [CHK07, DN07, Abu13]. We give a direct (high-level) description without using the abstraction by TDFs for clarity.

$\mathsf{Setup}(1^\kappa)$: Output a CRS crs consisting of a group description $(\mathbb{G}, p, g)$ and random group elements $(X_1, ..., X_\ell) \xleftarrow{\$} \mathbb{G}^\ell$ where ℓ is the length of the hidden random string of the underlying HBM-NIZK.

$\mathsf{Prove}(\mathsf{crs}, x, w)$: The prover samples $\tau \xleftarrow{\$} \mathbb{Z}_p$, computes $Z_i := X_i^\tau$ and lets ρ_i be the hardcore bit of Z_i for all $i \in [\ell]$. Then it uses $\rho := \rho_1 \| \cdots \| \rho_\ell$ as a hidden random string to generate a proof π_{hbm} along with an index set $I \subset [\ell]$ by the proving algorithm of the underlying HBM-NIZK on (x, w). It outputs a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{Z_i\}_{i \in I}, Y := g^\tau)$.

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$ Given a statement x and a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{Z_i\}_{i \in I}, Y := g^\tau)$, the verification algorithm verifies (g, X_i, Y, Z_i) is a DH-tuple for all $i \in I$ by using pairing, and rejects if it is not the case. Then it computes the hardcore bit ρ_i of Z_i for all $i \in I$, and verifies π_{hbm} by the verification algorithm of the underlying HBM-NIZK.

Roughly speaking, soundness and zero-knowledge follow from those of the underlying HBM-NIZK since a hidden random string ρ is somehow "committed" in $(X_1, ..., X_\ell)$ once τ is fixed, and only the sub-string of them corresponding to I is revealed to the verifier.[3] Clearly, the above construction relies on pairing to check if (g, X_i, Y, Z_i) is a DH-tuple

[3]Though a cheating prover can arbitrarily choose $\tau \in \mathbb{Z}_p$, we can negligibly bound its success probability by the union bound if the success probability of a cheating prover of the underlying HBM-NIZK is bounded by $p^{-1} \cdot \mathsf{negl}(\kappa)$.

during verification. We note that this check is essential since without it, a cheating prover can arbitrarily choose Z_i for $i \in I$ to control $\rho_{|I}$ to any value, in which case soundness of HBM-NIZK ensures nothing.

Getting rid of pairing. Now, we explain how to get rid of the use of pairing from the above construction in the DV setting. Our main idea is to use the twin-DH technique [CKS09]. Intuitively, the twin-DH technique enables a designated entity to verify whether a tuple $(g, X, Y, Z) \in \mathbb{G}^4$ is a DH-tuple without knowing the discrete logarithm of X or Y and without using pairings, where (g, X) is public, and (Y, Z) may be chosen arbitrarily. More precisely, suppose that an extra element $\widehat{X} := g^{\beta}/X^{\alpha}$ is published in addition to (g, X) where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$. Then for $Y = g^{\tau}$, we may consider $(Z = X^{\tau}, \widehat{Z} = \widehat{X}^{\tau})$ to be a "proof" that (g, X, Y, Z) is a DH-tuple. Namely, a designated verifier who holds α and β can verify the validity of the "proof" by checking if $Z^{\alpha}\widehat{Z} = Y^{\beta}$ holds. The main implication of the twin-DH technique is that the above verification is essentially equivalent to checking if $Z = X^{\tau}$ and $\widehat{Z} = \widehat{X}^{\tau}$ hold conditioned on the fact that $(Y, Z, \widehat{Z})$ is chosen by a "prover" who does not know $(\alpha.\beta)$.

With this technique in hand, we describe how to modify the above construction to achieve DV-NIZK without pairing: We add extra elements $\widehat{X}_i := g^{\beta_i}/X^{\alpha_i}$ where $\alpha_i, \beta_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [\ell]$ in the CRS, give $\{\alpha_i, \beta_i\}_{i\in[\ell]}$ as the verification key to the designated verifier, and add extra elements $\widehat{Z}_i := \widehat{X}_i^{\tau}$ for $i \in I$ in the proof. Then the verifier can verify that (g, X_i, Y, Z_i) is a DH-tuple by checking if $Z^{\alpha_i}\widehat{Z} = Y^{\beta_i}$ holds *without using pairing*. This enables us to achieve DV-NIZK without pairing.

On adaptive zero-knowledge. Though our main idea is as presented above, the above described construction only achieves *non-adaptive zero-knowledge* which requires an adversary to choose the statement x independently of the CRS. To achieve adaptive zero-knowledge, we need to add some extra structures using the technique of non-committing encryption [CFGN96, DN07]. See Section 3 for technical details. We note that the original FLS NIZK proof system is also adaptive zero-knowledge, but it uses specific properties of the underlying HBM-NIZK. Though a similar analysis may also yield alternative construction of DV-NIZKs with adaptive zero-knowledge from the CDH assumption without pairing, we choose the above approach where we do not assume any structure on the underlying HBM-NIZK for a conceptually simpler and modular construction.

Part 2: PP-NIZK via context-hiding HomAuth. Kim and Wu [KW18a] showed a conversion from any context-hiding HomAuth scheme to PP-NIZKs. In particular, they noted that context-hiding HomAuth scheme for $\mathbf{NC}^1$ suffices to instantiate their conversion. In this part, we propose new constructions of context-hiding HomAuth schemes for $\mathbf{NC}^1$, and plug them into their conversion. First, we recall the definition of HomAuth. Roughly speaking, a HomAuth scheme is a digital signature or MAC scheme with a homomorphic property. Namely, given a vector of signatures $\boldsymbol{\sigma}$ for a vector of messages $\mathbf{x}$, anyone can publicly evaluate the signature on a circuit C to generate an evaluated signature σ for a message $C(\mathbf{x})$. We say that a HomAuth scheme is a HomSig scheme if verification can be done publicly, and is a HomMAC scheme otherwise. As a security requirement of HomAuth scheme, we require that an adversary given $\mathbf{x}$ cannot generate a pair of an evaluated signature σ^* and a circuit C^* such that σ^* is a valid signature for a message $z \neq C^*(\mathbf{x})$ even if the adversary is given access to a verification oracle. In addition, we say that a HomAuth scheme is context-hiding if σ for a message z generated by evaluating a circuit C on a vector of signatures $\boldsymbol{\sigma}$ for $\mathbf{x}$ does not reveal information of $\mathbf{x}$ beyond that $C(\mathbf{x}) = z$.

In this paper, we propose two new constructions of HomAuth schemes for $\mathbf{NC}^1$. The first one is a HomSig scheme based on a new assumption that we call (n, m)- CDHER assumption on a pairing-group. A nice feature of this HomSig scheme is that the size of an evaluated signature is compact (i.e., does not depend on the message vector length or the circuit to evaluate), and has online-offline efficiency. The second one is a HomMAC scheme based on the DDH assumption on a pairing-free group. The function class the second scheme supports is arithmetic circuits over $\mathbb{Z}_p$ of polynomial degree, which is larger than $\mathbf{NC}^1$, and we take advantage of this extra freedom to improve the proof size. We explain these constructions below.

HomSig from CDHER. Here, we informally explain how an attribute-based encryption (ABE) scheme with some special properties can be converted into a HomSig scheme. Our HomSig scheme from the CDHER assumption can be seen as an instantiation of this conversion.

To explain the idea, we first recall the notion of (key-policy) ABE. In an ABE scheme, one can encrypt a message M with respect to some string $\mathbf{x} \in \{0,1\}^\ell$ using some public parameter pp. Furthermore, a secret key is associated with some policy $C : \{0,1\}^\ell \to \{0,1\}$ and the decryption is possible if and only if $C(\mathbf{x}) = 1$. As for security, we require the selective *one-way* security. In a selective one-way security game, an adversary has to declare its target $\mathbf{x}^\star$ at the beginning of the game before seeing the public parameter pp. An adversary can further query secret keys for C such that $C(\mathbf{x}^\star) = 0$ unbounded polynomially many times throughout the game, and we require that an adversary given an encryption of a *random* message $\mathsf{M}^\star$ under the string $\mathbf{x}^\star$ cannot recover $\mathsf{M}^\star$.

We first observe that the security proofs for most selectively secure schemes such as those proposed in [SW05, GPSW06a, Wat11, GVW15a, BGG+14] can be abstracted in the following manner:[4] At the beginning of the game, the reduction algorithm is given a problem instance Ψ of some hard problem (e.g., the bilinear Diffie-Hellman problem). Then, it first runs the adversary to obtain the target $\mathbf{x}^\star$. Given Ψ and $\mathbf{x}^\star$, the reduction algorithm generates pp along with some simulation trapdoor $\mathsf{td}_{\mathbf{x}^\star}$. The reduction algorithm can perfectly simulate the game using $\mathsf{td}_{\mathbf{x}^\star}$. Namely, given $\mathsf{td}_{\mathbf{x}^\star}$, it can generate correctly distributed secret key sk_C for any C such that $C(\mathbf{x}^\star) = 0$. Furthermore, given $\mathsf{td}_{\mathbf{x}^\star}$, it can embed the problem instance Ψ into the challenge ciphertext so that it can extract the answer of the hard problem whenever the adversary succeeds in extracting $\mathsf{M}^\star$.

Our basic idea for constructing HomSig is to use the above reduction algorithm in the real world. To sign on a message $\mathbf{x}$, we generate $\mathsf{td}_{\mathbf{x}}$ and set $\boldsymbol{\sigma} := \mathsf{td}_{\mathbf{x}}$. To evaluate the signature $\boldsymbol{\sigma}$ on a circuit C such that $C(\mathbf{x}) = 0$, we run the reduction algorithm of the ABE scheme on input $\mathsf{td}_{\mathbf{x}}$ to generate sk_C and set $\sigma := \mathsf{sk}_C$. Here, evaluation of signatures can be done publicly since $\mathsf{td}_{\mathbf{x}}$ is the only secret state required to run the reduction algorithm. A subtle problem with this approach is that we cannot evaluate the signature on a circuit C such that $C(\mathbf{x}) = 1$ since the reduction algorithm does not work for such C. This problem can be easily fixed by defining the scheme so that when evaluating a signature on such C, we generate $\mathsf{sk}_{\neg C}$ instead of sk_C, where $\neg C$ is a circuit that is obtained by flipping the output bit of C by applying the NOT gate. Now, for the signature $\sigma = \mathsf{sk}_C$ to be publicly verifiable, we require it to be possible to efficiently check whether σ is a correctly generated secret key of the ABE given (C, σ). However, this is not such a strong restriction since it is satisfied by many selectively secure ABE schemes such as the ones listed above.

We recall that given $\mathsf{td}_{\mathbf{x}}$, the reduction algorithm can perfectly simulate the selective security game for ABE where $\mathbf{x}$ is the target chosen by the adversary. This in particular implies that sk_C simulated by $\mathsf{td}_{\mathbf{x}}$ follows the same distribution as sk_C generated in the real system which does not use information of $\mathbf{x}$. Then, the context-hiding property of the scheme follows from this fact. Namely, the distribution of $\sigma = \mathsf{sk}_C$ only depends on C and pp, not on $\mathbf{x}$. In other words, σ does not leak any information of $\mathbf{x}$, which meets the requirements of the context-hiding security. Furthermore, the unforgeability of the scheme follows from the one-wayness of the ABE: If the adversary can forge a signature $\sigma = \mathsf{sk}_{C^\star}$ for $C^\star$ such that $C^\star(x) = 1$, then $\mathsf{sk}_{C^\star}$ can be used to decrypt the challenge ciphertext, which contradicts the security of the ABE. We note that the circuit class of the allowed homomorphic evaluation for the resulting HomSig scheme is roughly the same as the circuit class supported by the original ABE scheme.

In order to obtain the aforementioned HomSig scheme for $\mathbf{NC}^1$ with compact signatures, we need a key-policy ABE scheme with constant-size secret keys. Unfortunately, the only construction of ABE scheme [AHY15] which meets the efficiency (i.e., compactness) property we require does not conform to our template that uses the simulation trapdoor $\mathsf{td}_{\mathbf{x}}$. Therefore, we construct a new ABE scheme with the required property which conforms to our template based on the CDHER assumption. The structure of our ABE scheme is inspired by the ciphertext-policy ABE scheme with constant-size ciphertexts (*not* secret keys) due to Agrawal and Chase [AC16]. To turn their scheme into an ABE scheme with constant-size secret keys, at a high level, we swap the ciphertexts and secret keys of their construction. Since the security of the resulting scheme is not guaranteed by that of the original one, we directly prove its security by adding considerable modification to the previous proof techniques [RW13, AC17].

HomMAC from DDH. Here, we explain the construction of HomMAC under the DDH assumption. Our idea is to add the context-hiding property to the non-context-hiding HomMAC proposed by Catalano and Fiore [CF18] by using functional encryption for inner products (IPFE). First, we recall their non-context-hiding HomMAC, which supports all arithmetic circuits of polynomially bounded degree.[5] The signing/verification key of their construction are $\mathbf{r} \in \mathbb{Z}_p^\ell$

[4] Actually, these previous works prove the standard indistinguishability security notion rather than one-wayness. However, one-wayness is sufficient for our application.

[5] Though the original construction by Catalano and Fiore [CF18] is based on PRF, we present an information theoretically secure variant of it in a simplified setting where the arity of an arithmetic circuit is bounded.

and $s \in \mathbb{Z}_p^*$ where ℓ is the arity of arithmetic circuits it supports, and the evaluation key is a prime p. A signature $\boldsymbol{\sigma} \in \mathbb{Z}_p^\ell$ for a message $\mathbf{x} \in \mathbb{Z}_p^\ell$ is set to be $\boldsymbol{\sigma} := (\mathbf{r} - \mathbf{x})s^{-1} \mod p$.[6] Given an arithmetic circuit f of degree D, a message $\mathbf{x}$, and a signature $\boldsymbol{\sigma}$, the evaluation algorithm computes the coefficients $(c_1, ..., c_D) \in \mathbb{Z}_p^D$ that satisfy $f(\mathbf{r}) = f(\mathbf{x}) + \sum_{j=1}^{D} c_j s^j$, and sets $\sigma := (c_1, ..., c_D)$ as an evaluated signature. We remark that this can be done by using $\mathbf{x}$, $\boldsymbol{\sigma}$, and p without knowing $(r_1, ..., r_n)$ or s since the signatures satisfy $s\boldsymbol{\sigma} + \mathbf{x} = \mathbf{r} \mod p$. To verify the evaluated signature, the verifier simply checks if the above equation holds by using $\mathbf{r}$ and s included in the verification key. Though the construction is very simple, the scheme satisfies unforgeability even against unbounded-time adversaries. Unfortunately, this construction cannot yet be used for the purpose of PP-NIZKs, since in general it is not context-hiding.

Here, we observe that in the above construction, what a verifier has to know for the verification is only $\sum_{j=1}^{D} c_j s^j$, and not the entire $(c_1, ..., c_D)$. Moreover, $\sum_{j=1}^{D} c_j s^j$ does not convey any information on $\mathbf{x}$ beyond $f(\mathbf{x})$ because the term is determined solely by $\mathbf{r}$ and $f(\mathbf{x})$. Therefore if there exists a way to only transfer $\sum_{j=1}^{D} c_j s^j$ to the verifier, then context-hiding is guaranteed. We remark that a trivial idea of publishing s does not work because it completely breaks the unforgeability. In particular, we want to find a way to let a verifier only know $\sum_{j=1}^{D} c_j s^j$ without providing s to the evaluator. To solve this problem we rely on IPFE. In an IPFE scheme, both a ciphertext and a secret key are associated with a vector. If we decrypt a ciphertext of a vector $\mathbf{x}$ by a secret key associated with $\mathbf{y}$, then the decryption result is $\langle \mathbf{x}, \mathbf{y} \rangle$, which is an inner product of $\mathbf{x}$ and $\mathbf{y}$. We convert the above non-context-hiding HomMAC to a context-hiding one by using IPFE as follows: In the setup, we additionally generate a public parameter pp and a master secret key msk of IPFE. Then a verifier is provided with a secret key $\mathsf{sk}_{(s,...,s^D)}$ for a vector $(s, ..., s^D)$, and an evaluator is provided with pp. The evaluator sets the evaluated signature to be an encryption ct of $(c_1, ..., c_D)$ instead of $(c_1, , , ., c_D)$ itself. Now, a verifier only learns $\sum_{j=1}^{D} c_j s^j$ due to the security of IPFE, and thus context-hiding is achieved.

Given the above overview, it may seem that any IPFE scheme suffices for the construction. Moreover, since only one secret key is needed in the construction, it seems that one-key IPFE suffices. Since there are constructions of one-key secure FE even for all circuits based on any PKE scheme [SS10, GVW12], one may think that we can implement the above construction based on any PKE scheme. However, this is in fact not the case because these FE schemes are malleable. Namely, the standard security notion of FE does not prevent a malicious encryptor from generating an invalid ciphertext. Put differently, the decryption result may be controlled. In the context of the above construction, the fact that an evaluator generates a ciphertext ct by the secret key $\mathsf{sk}_{(s,...,s^D)}$ that is decrypted to T does not necessarily mean that it knows $(c_1, ..., c_D)$ such that $\sum_{j=1}^{D} c_j s^j = T$. Therefore, although the construction seems to work, we cannot prove unforgeability of the above scheme. To solve this problem, we introduce a notion which we call *extractability* for IPFE. Extractability requires that for any (possibly malformed) ciphertext ct that is decrypted to T with a secret key sk associated with a vector $\mathbf{y}$, we can extract $\mathbf{x}$ such that $\langle \mathbf{x}, \mathbf{y} \rangle = T$ from ct. It is clear that the above problem is resolved if we have an extractable IPFE.

Here, we observe that the IPFE scheme based on the DDH assumption proposed by Agrawal, Libert, and Stehlé [ALS16] satisfies extractability. A subtle problem of their construction is that a decryptor must compute a discrete logarithm for computing a decryption result, and thus the size of the decryption result must be limited to being relatively small. Fortunately, this does not matter in our application since the verification is done by simply checking if a decryption result of IPFE satisfies a certain linear equation which can be performed on the exponent. Concretely, we only need a variant of IPFE that enables a decryptor to learn inner-product on the exponent. Putting all the ideas together, we obtain a context-hiding HomMAC for arithmetic circuits of polynomial degree (which includes $\mathbf{NC}^1$) based on the DDH assumption, which further combined with [KW18a] leads to PP-NIZK proofs based on the DDH assumption. Moreover, we can make the proof size of the PP-NIZK short by incorporating the idea by Katsumata [Kat17]. Namely, the proof size of the resulting PP-NIZK is $|C| + \mathsf{poly}(\kappa)$ where $|C|$ is the size of a circuit that computes a relation to prove. See Appendix C.3 for details.

PP-NIZK with sublinear proof size. Direct adaptations of the Kim-Wu conversion to compact context-hiding HomAuth for $\mathbf{NC}^1$ yield PP-NIZK with proof sizes $|C| + \mathsf{poly}(\kappa)$. Here, we explain that this can be further reduced to *sublinear size* $|w| + |C|/\log \kappa + \mathsf{poly}(\kappa)$ by making a slight relaxation that a circuit C computing the **NP** relation is expressed as a *leveled circuit* [BGI16]; a circuit whose gates are partitioned into $D+1$ levels and all incoming wires to a gate of level $i+1$ come from gates of level i for each $i \in [D]$. To explain this, we first briefly review the Kim-Wu conversion. In their

[6] Though the scheme is not publicly verifiable, we call $\boldsymbol{\sigma}$ a "signature" for compatibility to HomSig.

construction, a prover is provided with a secret key K of a symmetric key encryption (SKE) scheme as its proving key, and to prove that (x, w) satisfies $C(x, w) = 1$ for a circuit C, it encrypts w by using K to generate a ciphertext ct, and generates an evaluated signature σ on message "1" under the function $f_{\mathsf{ct},x}$ defined by $f_{\mathsf{ct},x}(K') := C(x, \mathsf{Dec}(K', \mathsf{ct}))$ where Dec is the decryption algorithm of the SKE scheme. A proof consists of ct and σ. A verifier simply verifies that the evaluated signature σ is a valid signature on message "1" under the function $f_{\mathsf{ct},x}$. To implement this construction based on HomAuth for $\mathbf{NC}^1$, we have to express a circuit that computes the **NP** relation in $\mathbf{NC}^1$. This is in general possible by "expanding" the witness to values corresponding to all wires of $C(x, \cdot)$. However, since the size of the expanded witness is as large as the circuit size $|C|$, the proof size of the resulting PP-NIZK is linear in $|C|$. Now, we observe that we actually need not expand the witness to all wires, and we can choose a portion of them based on a similar idea used in [BGI16]. Namely, for a leveled circuit C of depth D, we divide $[D]$ into $\log \kappa$ intervals of length $D/\log \kappa$, and choose "special levels" i in each interval so that the number of gates of level i is the smallest among those in the interval. Then we set an expanded witness to be the original witness appended by values corresponding to all wires *of special levels* of $C(x, \cdot)$. We observe that the consistency of the expanded witness generated in this way still can be verified in $\mathbf{NC}^1$ since successive special levels are at most $2 \log \kappa$ apart from each other. Moreover, the size of the expanded witness is at most $|w| + |C|/\log \kappa$ since the number of gates of special levels is at most $|C|/\log \kappa$ by the choice of special levels. Thus, by applying the Kim-Wu conversion with the above expanded witness, we obtain PP-NIZK with proof size $|w| + |C|/\log \kappa + \mathsf{poly}(\kappa)$.

1.4 Other Related Works

Concurrent Works. There are two concurrent and independent works [CH19, QRW19] that contain similar results to our first result, namely, multi-theorem DV-NIZK from CDH assumption in pairing-free groups. We summarize differences of these results below.

- Couteau and Hofheinz [CH19] additionally give a construction of (CRS,DV)-NIZK assuming the LWE assumption and a (CRS,DV)-non-interactive witness indistinguishable proof system for bounded distance decoding.
- Quach, Rothblum, and Wichs [QRW19] additionally consider a stronger variant of DV-NIZK called malicious DV-NIZK, and construct it based on a stronger assumption called the one-more CDH assumption in pairing-free groups.
- Constructions of (DP,PP)-NIZKs with compact proofs are unique to this paper.

CRS-NIZK from Lattices. Very recently, Peikert and Shiehian [PS19] constructed the first CRS-NIZKs for **NP** under standard lattice assumptions following the line of researches [KRR17, CCRR18, HL18, CCH+19] to instantiate the Fiat-Shamir transform [FS87] in the standard model.

More discussions on existing (DV, DP, PP)-NIZK. Unlike CRS-NIZKs where proving statements and verifying proofs can be done publicly, in (DV, DP, PP)-NIZKs since we have the notion of secret states, it is not uncommon to have a bound on the number of statements (i.e., theorems) one can prove without compromising soundness or zero-knowledge. In DV-NIZKs, a common issue have been the bound on the number of time the prover can query the verification oracle. Namely, a prover can break the soundness of a DV-NIZK if the verifier uses the same verification key to verify multiple statements. Due to this fact, such DV-NIZKs that require a bound on the number of time a prover can query the verification oracle are called *bounded-theorem*. If the verifier can keep using the same key for multiple statements, then it is called *multi-theorem*. Almost all previous DV-NIZKs for all of **NP** [PsV06, DFN06, VV09, CG15, Lip17] suffered from this issue of being bounded-theorem. There are more recent works that avoid the above issue based on a certain type of additively homomorphic encryption [CC18] or a primitive called oblivious linear-function evaluation [CDI+18]. However, instantiating either of these primitives require an assumption that is already known to imply a CRS-NIZK. DP and PP-NIZKs share similar problems, where in this case, zero-knowledge does not hold if the prover uses the same proving key multiple statements. Other than the recent schemes by Kim and Wu [KW18a] and Boyle et al [BCGI18], all previous DP or PP-NIZKs [DMP90, KMO90, LS91, Dam93, CD04, IKOS09] are known to be bounded-theorem. Though it is known that we can convert any bounded theorem NIZK to unbounded theorem NIZK in the CRS setting [FLS99], the conversion heavily relies on the fact that proofs can be generated publicly, and does not seem to work in the PP model. We refer to [KW18a] for more discussions.

Homomorphic authenticators. The notion of homomorphic authenticators (MACs or signatures) originates to Desmedt [Des93] and was first formalized by Johnson et al. [JMSW02]. In the beginning, HomAuth was considered extensively in the context of network coding where the homomorphism were focused on linear functions, yielding a long line of interesting works such as [AB09, BFKW09, GKKR10, BF11a, AL11, BF11b, CFW12, Fre12, CMP14, CFN15]. HomAuth for linear functions has also been considered for proofs of retrievability for outsourced storage [ABC+07, SW08]. Boneh and Freeman [BF11a] were the first to consider homomorphism beyond linear functions, showing the first scheme for polynomial function based on lattices. Since then numerous improvements on HomAuth have been made [CFW12, GW13, GVW15b, CF18]. Gorbunov et al. [GVW15b] constructed a HomSig that supports arbitrary circuits with bounded-depth from lattices and Catalano et al. [CF18] constructed a HomMAC that supports arbitrary arithmetic circuits with bounded-degree from PRFs or DH-type assumptions.

Recently, Tsabary [Tsa17] showed a generic conversion of an attribute-based signature (ABS) to HomSig. Using their construction, we may obtain a HomSig with compact signatures starting from an ABS with short signatures. However, the two ABS schemes with short signatures are not a complete fit for the conversion: The scheme by Attrapadung et al. [AHY15] is only selectively-secure and the above conversion is not applicable. The scheme by [NP15] is constructed on composite-order groups, which is not desirable from the view points of security and efficiency.

Finally, we also mention that our idea of viewing some types of ABE as HomSig seems to be applicable for other ABE schemes such as [GPSW06a]. This leads to a context-hiding HomSig scheme from the CDH assumption and thus DP-NIZK from the same assumption via the transformation due to Kim and Wu [KW18a]. In addition, we observe that if we start from the ABE for circuits from lattices due to Boneh et al. [BGG+14], we recover the existing HomSig scheme by Gorbunov, Vaikuntanathan, and Wichs [GVW15b]. While this is not a new result, the observation provides new insights into the connection between them.

2 Preliminaries

Notations. For a distribution or random variable X, we write $x \xleftarrow{\$} X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \xleftarrow{\$} S$ to denote the operation of sampling a random s from the uniform distribution over S.

2.1 Basic Notions

Here, we review some basic statistical notions.

Definition 2.1 (Statistical Distance). *Let $\mathcal{X}^{(b)} = \{X_\kappa^{(b)}\}_{\kappa\in\mathbb{N}}$ for $b \in \{0,1\}$ be two ensembles of random variables indexed by $\kappa \in \mathbb{N}$. The statistical distance between $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ over a countable set S is defined as $\Delta(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}) := \frac{1}{2}\sum_{\alpha\in S} |\Pr[X_\kappa^{(0)} = \alpha] - \Pr[X_\kappa^{(1)} = \alpha]|$. We say that $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ are statistically indistinguishable (denoted by $\mathcal{X}^{(0)} \overset{\text{stat}}{\approx} \mathcal{X}^{(1)}$) if $\Delta(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}) \le \mathsf{negl}(\kappa)$.*

Definition 2.2 (Min-entropy). *Let X be a random variable. The min-entropy of X is defined as follows.*

$$H_\infty(X) := \min_{x\in S}\{-\log \Pr[X = x]\} = -\log\left(\max_{x\in S} \Pr[X = x]\right).$$

Lemma 2.3 (Leftover Hash Lemma). *Let X be a random variable over $\mathcal{X}$ and $H_\infty(X) \ge k$. Fix $\epsilon > 0$. Let $\mathcal{H} : \mathcal{X} \to \{0,1\}^m$ be a universal hash family of size 2^u with output length $m = k - 2\log(1/\epsilon)$. We define $\mathsf{Ext}(X, h) := h(X)$ where $h \in \mathcal{H}$. Then, Ext is a strong $(k, \epsilon/2)$ extractor with seed length u and output length m. That is, for any random variable X on $\mathcal{X}$ independent of $h \xleftarrow{\$} \mathcal{H}$ with $H_\infty(X) \ge k$, it holds that $\Delta((h(X), h), (U_m, h)) \le \epsilon/2$ where U_m is the uniform distribution over $\{0,1\}^m$.*

Definition 2.4 (Pseudorandom Generators). *Let $n = n(\kappa)$ and $m = m(\kappa)$ be positive integer valued functions such that $m > n$. A function $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ is called a pseudorandom function if PRG is polynomial time computable and for every efficient algorithm $\mathcal{A}$ we have the following:*

$$\left|\Pr\left[x \xleftarrow{\$} \{0,1\}^n : 1 \leftarrow \mathcal{A}(1^\kappa, \mathsf{PRG}(x))\right] - \Pr\left[y \xleftarrow{\$} \{0,1\}^m : 1 \leftarrow \mathcal{A}(1^\kappa, y)\right]\right| \le \mathsf{negl}(\kappa).$$

We also introduce two basic hardness assumptions that will be respectively used in Section 3 and Section 5. Let GGen be a PPT algorithm that on input 1^κ returns a description $\mathcal{G} = (\mathbb{G}, p, g)$ where $\mathbb{G}$ is a cyclic group of prime order p and g is the generator of $\mathbb{G}$. Then the computational Diffie-Hellman assumption is defined as follows.

Definition 2.5 (Computational Diffie-Hellman Assumption). *We say that the computational Diffie-Hellman (CDH) assumption holds relative to* GGen *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\ \mathcal{G} = (\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa),\ \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p :\ g^{\alpha\beta} \leftarrow \mathcal{A}(1^\kappa, \mathcal{G}, g^\alpha, g^\beta)\ \right] \leq \mathsf{negl}(\kappa).$$

The decisional variant of the CDH assumption is defined as follows.

Definition 2.6 (Decisional Diffie-Hellman Assumption). *We say that the decisional Diffie-Hellman (DDH) assumption holds relative to* GGen *if for all PPT adversaries* $\mathcal{A}$*,*

$$\left|\Pr\left[\ \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p :\ 1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{G}, g^\alpha, g^\beta, g^{\alpha\beta})\ \right] - \Pr\left[\ \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p :\ 1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{G}, g^\alpha, g^\beta, g^\gamma)\ \right]\right| \leq \mathsf{negl}(\kappa),$$

where $\mathcal{G} = (\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.

2.2 Preprocessing NIZKs

Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomial time recognizable binary relation. For $(x, w) \in \mathcal{R}$, we call x as the statement and w as the witness. Let $\mathcal{L}$ be the corresponding **NP** language $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. We also write $\mathcal{R}(x, w) \in \{0, 1\}$ as the output of the polynomial time decision algorithm $\mathcal{R}$ on input (x, w), where 0 is for reject and 1 is for accept. Below, we define (adaptive multi-theorem) preprocessing NIZKs for **NP** languages. Some discussions on our presentation of NIZKs are provided below.

Definition 2.7 (NIZK Proofs). *A non-interactive zero-knowledge (NIZK) proof in the preprocessing model* Π_{PPNIZK} *for the relation* $\mathcal{R}$ *is defined by the following three polynomial time algorithms:*

$\mathsf{Setup}(1^\kappa) \to (\mathsf{crs}, k_\mathsf{P}, k_\mathsf{V})$*: The setup algorithm takes as input the security parameter* 1^κ *and outputs a common reference string* crs*, a proving key* k_P*, and a verification key* k_V*. This algorithm is executed as the "preprocessing" step.*

$\mathsf{Prove}(\mathsf{crs}, k_\mathsf{P}, x, w) \to \pi$*: The prover's algorithm takes as input a common reference string* crs*, a proving key* k_P*, a statement* x*, and a witness* w *and outputs a proof* π*.*

$\mathsf{Verify}(\mathsf{crs}, k_\mathsf{V}, x, \pi) \to \top$ *or* $\bot$*: The verifier's algorithm takes as input a common reference string, a verification key* k_V*, a statement* x*, and a proof* π *and outputs* $\top$ *to indicate acceptance of the proof and* $\bot$ *otherwise.*

Moreover, an (adaptive multi-theorem) NIZK proof in the preprocessing model Π_{PPNIZK} *is required to satisfy the following properties, where the probabilities are taken over the random choice of the algorithms:*

Completeness. *For all pairs* $(x, w) \in \mathcal{R}$*, if we run* $(\mathsf{crs}, k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(1^\kappa)$*, then we have*

$$\Pr[\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, k_\mathsf{P}, x, w) : \mathsf{Verify}(\mathsf{crs}, k_\mathsf{V}, x, \pi) = \top] = 1.$$

Soundness. *For all (possibly inefficient) adversaries* $\mathcal{A}$*, if we run* $(\mathsf{crs}, k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(1^\kappa)$*, then we have*

$$\Pr[(x, \pi) \leftarrow \mathcal{A}^{\mathsf{Verify}(\mathsf{crs}, k_\mathsf{V}, \cdot, \cdot)}(1^\kappa, \mathsf{crs}, k_\mathsf{P}) : x \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, k_\mathsf{V}, x, \pi) = \top] = \mathsf{negl}(\kappa).$$

Here, in case soundness only holds for computationally bounded adversaries $\mathcal{A}$*, we say it is a NIZK* argument*.*

(Non-Programmable CRS) Zero-Knowledge. *For all PPT adversaries* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that if we run* $(\mathsf{crs}, k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(1^\kappa)$ *and* $\tau_\mathsf{V} \leftarrow \mathcal{S}_1(1^\kappa, \mathsf{crs}, k_\mathsf{V})$*, then we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs}, k_\mathsf{P}, \cdot, \cdot)}(1^\kappa, \mathsf{crs}, k_\mathsf{V}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\mathsf{crs}, k_\mathsf{V}, \tau_\mathsf{V}, \cdot, \cdot)}(1^\kappa, \mathsf{crs}, k_\mathsf{V}) = 1]\right| = \mathsf{negl}(\kappa),$$

where $\mathcal{O}_0(\mathsf{crs}, k_\mathsf{P}, x, w)$ *outputs* $\mathsf{Prove}(\mathsf{crs}, k_\mathsf{P}, x, w)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise, and* $\mathcal{O}_1(\mathsf{crs}, k_\mathsf{V}, \tau_\mathsf{V}, x, w)$ *outputs* $\mathcal{S}_2(\mathsf{crs}, k_\mathsf{V}, \tau_\mathsf{V}, x)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise.*

Remark 2.8 (Programmable Zero-Knowledge). As also discussed in [KW18b], we can define a slightly weaker variant of zero-knowledge where the simulator is provided the freedom of programming the common reference string crs and verification key k_{V}.

(Programmable CRS) Zero-Knowledge For all PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that if we run $(\mathsf{crs}, k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $(\overline{\mathsf{crs}}, \bar{k}_{\mathsf{V}}, \bar{\tau}_{\mathsf{V}}) \leftarrow \mathcal{S}_1(1^\kappa)$, then we have

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs}, k_{\mathsf{P}}, \cdot, \cdot)}(1^\kappa, \mathsf{crs}, k_{\mathsf{V}}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\overline{\mathsf{crs}}, \bar{k}_{\mathsf{V}}, \bar{\tau}_{\mathsf{V}}, \cdot, \cdot)}(1^\kappa, \overline{\mathsf{crs}}, \bar{k}_{\mathsf{V}}) = 1]\right| = \mathsf{negl}(\kappa),$$

where $\mathcal{O}_0(\mathsf{crs}, k_{\mathsf{P}}, x, w)$ outputs $\mathsf{Prove}(\mathsf{crs}, k_{\mathsf{P}}, x, w)$ if $(x, w) \in \mathcal{R}$ and $\perp$ otherwise, and $\mathcal{O}_1(\overline{\mathsf{crs}}, \bar{k}_{\mathsf{V}}, \bar{\tau}_{\mathsf{V}}, x, w)$ outputs $\mathcal{S}_2(\overline{\mathsf{crs}}, \bar{k}_{\mathsf{V}}, \bar{\tau}_{\mathsf{V}}, x)$ if $(x, w) \in \mathcal{R}$ and $\perp$ otherwise.

This definition captures the zero-knowledge property used in standard NIZKs in the common reference string (CRS) model. In the CRS model, the Setup algorithm outputs a CRS σ used by both the prover and verifier, and the zero-knowledge simulator is allowed to program the CRS σ. Specifically, the proving key and verification key are both set as the CRS σ.

Remark 2.9 (Different types of NIZKs). The definition is general enough to capture many of the existing types of NIZKs. In case $k_{\mathsf{P}} = k_{\mathsf{V}} = \perp$, the above definition captures the standard NIZKs in the common reference string (CRS) model, which we refer to as CRS-NIZKs hereafter. Specifically anybody can construct a proof using the public CRS and those proofs are publicly verifiable [FLS99]. On the other hand, in case $k_{\mathsf{P}} = \perp$ but k_{V} is required to be kept secret, the above definition captures *designated verifier* NIZKs (DV-NIZKs) [PsV06, DFN06]. Moreover, in case $k_{\mathsf{V}} = \perp$ but k_{P} is required to be kept secret, the above definition captures designated prover NIZKs (DP-NIZKs) [KW18a]. Finally, in case both k_{P} and k_{V} must be kept secret, it is simply called preprocessing NIZKs (PP-NIZKs) [CD04].

Remark 2.10 (Adaptive and Non-Adaptive NIZK). One often considers weaker security called *non-adaptive* soundness and zero-knowledge. In non-adaptive soundness, an adversary has to declare the statement x on which he forges a proof before seeing a common reference string. In non-adaptive zero-knowledge, an adversary has to declare a pair of a statement x and its witness w to query the proving oracle before seeing a common reference string. All the NIZKs we construct in this paper satisfy adaptive soundness and zero-knowledge.

NIZKs for Bounded Languages. Throughout this paper, we mainly consider the weaker variant of PP-NIZKs which we call PP-NIZKs *for bounded languages* as was done by Kim and Wu [KW18a]. PP-NIZKs for bounded languages enable one to generate a proof for $(x, w) \in \mathcal{R} \cap (\{0,1\}^{n(\kappa)} \times \{0,1\}^{m(\kappa)})$ for a priori bounded polynomials $n(\cdot)$ and $m(\cdot)$. For clarity, we say PP-NIZKs *for unbounded languages* to express PP-NIZKs that do not have the above limitation. As discussed in Appendix C.5, we can generically convert any PP-NIZKs for bounded languages to PP-NIZKs for unbounded languages at the cost of making the proof size larger. However, we note that since the conversion makes the proof size larger, the distinction between PP-NIZKs for bounded and unbounded languages are meaningful if we start to consider proof sizes.

2.3 (Simplified) Homomorphic Authenticators

We provide the definition of a simplified homomorphic authentication (HomAuth) scheme with the minimal property required for the NIZK construction. Unlike standard HomAuth schemes, in our definition succinctness or efficient verification will not be a requisite; the essential property we require is *context-hiding*. Some discussions on the reason behind the choice of our definition are provided below. The more standard definition of HomAuth schemes is provided in Appendix D.

Syntax. Let $\{\mathcal{X}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of message spaces. Let $\{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of circuits, where $\mathcal{C}_\kappa$ is a set of polynomial sized circuits with domain $\mathcal{X}_\kappa^{\ell(\kappa)}$ and range $\mathcal{X}_\kappa$ whose depth is bounded by $d(\kappa)$. In the following, we occasionally drop the subscripts when the meaning is clear. In case $\mathcal{X} = \{0,1\}$, we set $\mathcal{C}$ to be simply a family of boolean circuits.

Below, we define a *single-shot* homomorphic authenticator scheme, where the signing algorithm is defined to run against all the messages $\mathbf{x} \in \mathcal{X}^\ell$ at once. Furthermore, the verification algorithm for fresh (i.e., non-evaluated)

signatures will be verified all at once as well. Let $\{\Sigma_{\mathsf{Fresh}_\kappa}\}_{\kappa\in\mathbb{N}}$ and $\{\Sigma_{\mathsf{Evaled}_\kappa}\}_{\kappa\in\mathbb{N}}$ be families of signature spaces, where each of them corresponds to the output space of fresh signatures and evaluated signatures, respectively.

Definition 2.11 (Homomorphic Authenticators). *A homomorphic authenticator (HomAuth) scheme Π_{HA} with message space $\mathcal{X}$ for the circuit class $\mathcal{C}$ is defined by the following five algorithms:*

$\mathsf{HA.KeyGen}(1^\kappa, 1^\ell) \to (\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$*: The key generation algorithm takes as input the security parameter 1^κ and the message length 1^ℓ and outputs a verification key* vk*, an evaluation key* ek*, and a signing key* sk*.*

$\mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x} = (x_1, \cdots, x_\ell)) \to \boldsymbol{\sigma}$*: The signing algorithm takes as input a signing key* sk *and messages $\mathbf{x} \in \mathcal{X}^\ell$, and outputs a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$.*

$\mathsf{HA.Eval}(\mathsf{ek}, C, \mathbf{x}, \boldsymbol{\sigma}) \to \sigma$*: The signature-evaluation algorithm takes as input an evaluation key* ek*, a circuit $C : \mathcal{X}^\ell \to \mathcal{X}$ in $\mathcal{C}$, messages $\mathbf{x} \in \mathcal{X}^\ell$, and a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$ and outputs an evaluated signature $\sigma \in \Sigma_{\mathsf{Evaled}}$.*

$\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma}) \to \top$ *or* $\bot$*: The fresh verification algorithm takes as input a verification key* vk*, messages $\mathbf{x} \in \mathcal{X}^\ell$, and a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$, and outputs $\top$ if the signature is valid and outputs $\bot$ otherwise.*

$\mathsf{HA.VerifyEvaled}(\mathsf{vk}, C, z, \sigma) \to \top$ *or* $\bot$*: The evaluated verification algorithm takes as input the verification key* vk*, a circuit $C \in \mathcal{C}$, a message $z \in \mathcal{X}$, and a signature $\sigma \in \Sigma_{\mathsf{Evaled}}$, and outputs $\top$ if the signature is valid and outputs $\bot$ otherwise.*

Publicly Verifiable Authenticators: *If the scheme supports public verifiability, i.e., the verification key* vk *can be made public, then we call the above scheme a* homomorphic signature (HomSig) *scheme. In this case, we may include the evaluation key* ek *in the verification key* vk*, without loss of generality, and replace* ek *with* vk *in the above syntax.*

Designated Verifier Authenticators: *If the scheme only supports private verification, i.e., the verification key* vk *must be kept secret, then we call the above scheme a* homomorphic MAC (HomMAC) *scheme.*

Remark 2.12 (Efficient Verification). Our syntax dismisses two algorithms that are typically present in standard HomAuth formalization. The two dismissed algorithms are meaningful for two types of efficient verification: online-offline efficiency and amortized efficiency.

Online-Offline Efficiency. In standard HomAuth definitions, there is an extra algorithm called the verification key (or public key) evaluation algorithm which takes as input the verification key vk (or a public key pk) and a circuit C, and outputs an evaluated verification key vk_C (or evaluated public key pk_C). The verifier can then use vk_C to verify a message-signature pair (z, σ_C) were σ_C is a signature evaluated on the circuit C. In case the verification time of the message-signature pair (z, σ_C) is $\mathsf{poly}(\lambda)$ (resp. $\mathsf{poly}(\lambda, \log|C|)$), i.e., the verification time is strictly less then the computation time of C, we say the HomAuth scheme has online-offline (resp. weak online-offline) efficiency; the verifier can prepare the evaluated verification key vk_C in the offline phase without knowledge of the yet to be known message z and use the preprocessed vk_C during the online phase. Although this property is not essential when constructing NIZKs, it is nonetheless an appealing notion for an HomAuth scheme to have. In Appendix D we provide a more formal definition of the this notion and show that one of our HomAuth construction satisfies them in the *multi-data* setting.

Amortized Efficiency Over Datasets Signed by Different Signing Keys. In addition, our syntax does not include the so-called parameter generation algorithm, which is used to output a public key pk (also termed as the public parameters param in some literatures) that is an independent component from the verification, evaluation, and signing keys $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$. In standard HomAuth context, several keys $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$ may be associated with a single public key pk. Combined with the above public key evaluation algorithm, this has the effect of amortized verification of a computation C over datasets signed by different users. In particular, if pk_C is created once in the offline phase, it can be used to verify any message-signature pair (z, σ_C) created by any signing key sk in an efficient manner. As above, we omit this notion since it is not required for constructing NIZKs. We note that none of our HomAuth constructions are proven to have amortized efficiency.

Remark 2.13 (Fresh and Evaluated Signature Spaces). In many of the previous leveled HomAuth schemes, the structure of the output of Σ_{Fresh} and Σ_{Evaled} were the same. Therefore, there was no need to explicitly define two different verification algorithms $\mathsf{HA.VerifyFresh}$ and $\mathsf{HA.VerifyEvaled}$. However, since our HomAuth schemes output a fresh and evaluated signatures with different structures, it will be convenient to define the two verification algorithms.

Remark 2.14 (Single-Shot Signing and Verification Algorithm). We restrict the signing algorithm $\mathsf{HA.Sign}$ and the fresh verification algorithm $\mathsf{HA.VerifyFresh}$ to sign the messages and verify all the fresh signatures in one-shot, respectively. We note that our HomAuth construction in Section 5 satisfies the standard notion where the signing and fresh verification algorithms are run for independent messages and signatures.

Correctness. There are two types of correctness which a HomAuth scheme must satisfy: signing correctness and evaluation correctness. Formally, they are defined as follows:

Definition 2.15 (Correctness). *We say a homomorphic authentication scheme Π_{HA} is correct, if for all $\kappa \in \mathbb{N}$, $\ell \in \mathsf{poly}(\kappa)$, messages $\mathbf{x} = (x_1, \cdots, x_\ell) \in \mathcal{X}^\ell$, and $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \in \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$ the following two conditions hold:*
(1) Signing Correctness: *For all $\boldsymbol{\sigma} \in \mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x})$ in Σ_{Fresh}, we have*

$$\Pr[\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma}) = \top] = 1.$$

(2) Evaluation Correctness: *For all circuits $C \in \mathcal{C}$, signatures $\boldsymbol{\sigma}$ such that $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma}) = \top$, and $\sigma \in \mathsf{HA.Eval}(\mathsf{ek}, C, \mathbf{x}, \boldsymbol{\sigma})$ in Σ_{Evaled}, we have*

$$\Pr[\mathsf{HA.VerifyEvaled}(\mathsf{vk}, C, C(\mathbf{x}), \sigma) = \top] = 1.$$

(Single-Shot) Unforgeability. We now define *single-shot* unforgeability for a HomAuth scheme, where the adversary must declare the challenge messages all at once. Below we assume that checking membership of $\mathcal{C}, \Sigma_{\mathsf{Fresh}}, \Sigma_{\mathsf{Evaled}}$ can be done efficiently. The security notion is defined formally by the following game between a challenger and an adversary $\mathcal{A}$.

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given 1^κ as input and sends 1^ℓ to the challenger. Then the challenger generates a signing-verification key pair $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$ and gives ek to $\mathcal{A}$. In case it is an HomSig scheme, i.e., the signatures are publicly verifiable, then the challenger also provides vk to $\mathcal{A}$.

Signing Query: The adversary $\mathcal{A}$ submits a set of messages $\mathbf{x} \in \mathcal{X}^\ell$ to be signed. The challenger responds by creating a signature $\boldsymbol{\sigma} \leftarrow \mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x})$ and sends $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$ to $\mathcal{A}$. Here, $\mathcal{A}$ can query a set of messages only once.

Verification Query: The adversary $\mathcal{A}$ may adaptively query a message-signature(-circuit) pair. When the query is of type $(\mathbf{x}, \boldsymbol{\sigma}) \in \mathcal{X}^\ell \times \Sigma_{\mathsf{Fresh}}$, the challenger returns the output of $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma})$. When the query is of type $(z, \sigma, C) \in \mathcal{X} \times \Sigma_{\mathsf{Evaled}} \times \mathcal{C}$, the challenger returns the output of $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, C, z, \sigma)$. For any other types of queries, the challenger returns $\perp$.[7]

Forgery: Then the adversary $\mathcal{A}$ outputs a circuit $C^\star$, a message $z^\star \in \mathcal{X}$, and a signature $\sigma^\star$ as the forgery. We say that $\mathcal{A}$ wins the game if:

1. $C^\star \in \mathcal{C}$ and $\sigma^\star \in \Sigma_{\mathsf{Evaled}}$;
2. $C^\star(\mathbf{x}) \neq z^\star$; and
3. $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, C, z^\star, \sigma^\star) = \top$.

The advantage of an adversary winning the above game is defined by $\Pr[\mathcal{A} \text{ wins}]$, where the probability is taken over the randomness used by the challenger and the adversary.

[7] Note that if we consider an HomSig scheme, this item can be dismissed since the verification step can be executed by the adversary himself.

Definition 2.16 ((Single-Shot) Unforgeability). *A homomorphic authenticator scheme* Π_{HA} *is said to satisfy (single-shot)* statistical *unforgeability if for any (possibly inefficient) adversary* $\mathcal{A}$ *the advantage* $\Pr[\mathcal{A}$ *wins*$]$ *of the above game is negligible. In case it only holds for adversaries that are computationally bounded, we say it satisfies* computational *unforgeability.*

We say Π_{HS} satisfies a weaker notion of *selective* (single-shot) unforgeability in case no adversary $\mathcal{A}$ that commits to the challenge messages $\mathbf{x}$ before seeing the evaluation key ek (and also the verification key vk in the case of HomSig schemes) can win the above game with more than negligible probability.

Remark 2.17 (Stronger Unforgeability). Instead of the single-shot security notion where the adversary must query all the messages in one shot, one can consider a stronger security notion where the adversary can adaptively query the messages [Fre12, GW13, CFN18]. However, we keep the security notion simple in our work, since we only require the single-shot notion for our generic construction of NIZK arguments.

Context-Hiding. We now define context-hiding for a HomAuth scheme. This security notion roughly states that an evaluated signature σ_C does not leak any information of the initial messages $\mathbf{x}$ other than the value $C(\mathbf{x})$. We first provide the *computational* variant of context-hiding. The notion is defined formally by the following game between an adversary $\mathcal{A}$ and a challenger running a simulator $\mathsf{HA.Sim}$.

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given 1^κ as input and sends 1^ℓ to the challenger. Then the challenger generates a signing-verification key pair $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$ and gives $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$ to $\mathcal{A}$.

Signing Query: The adversary $\mathcal{A}$ submits a circuit $C \in \mathcal{C}$, messages $\mathbf{x} \in \mathcal{X}^\ell$ and signatures $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$. The challenger returns $\perp$ if $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma}) = \perp$. Otherwise it generates $\sigma^{(0)} \leftarrow \mathsf{HA.Eval}(\mathsf{ek}, C, \mathbf{x}, \boldsymbol{\sigma})$ and $\sigma^{(1)} \leftarrow \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, C, C(\mathbf{x}))$, picks $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and returns $\sigma^{(\mathsf{coin})}$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs coin' as its guess for coin. We say that $\mathcal{A}$ wins the game if $\mathsf{coin}' = \mathsf{coin}$.

The advantage of an adversary wining the above game w.r.t a simulator $\mathsf{HA.Sim}$ is defined by $|\Pr[\mathcal{A} \text{ wins}] - 1/2|$.

Definition 2.18 (Computational Context-Hiding). *A homomorphic authenticator scheme* Π_{HA} *is computationally context-hiding if there exists a PPT simulator* $\mathsf{HA.Sim}$ *such that for any computationally bounded adversary* $\mathcal{A}$ *the advantage* $|\Pr[\mathcal{A}$ *wins*$] - 1/2|$ *of the above game is negligible.*

We also provide a simulation-based notion of security following the work of [GVW15b], which captures a stronger *statistical* notion of context-hiding.

Definition 2.19 (Statistical Context-Hiding). *A homomorphic authenticator scheme* Π_{HA} *is statistically context-hiding if for all* $\kappa \in \mathbb{N}$, $\ell \in \mathsf{poly}(\kappa)$, *there exists a PPT simulator* $\mathsf{HA.Sim}$ *such that, for any* $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \in \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$, $C \in \mathcal{C}$, *any pair* $(\mathbf{x}, z) \in \{(\mathbf{x}, z) \in \mathcal{X}^\ell \times \mathcal{X} \mid C(\mathbf{x}) = z\}$, *and* $\boldsymbol{\sigma} \in \mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x})$, *we have*

$$\{\sigma \leftarrow \mathsf{HA.Eval}(\mathsf{ek}, C, \mathbf{x}, \boldsymbol{\sigma})\} \overset{\text{stat}}{\approx} \{\sigma \leftarrow \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, C, z)\},$$

where the probability is only over the randomness used by the algorithms $\mathsf{HA.Eval}$ *and* $\mathsf{HA.Sim}$.

It is easy to see that statistical context-hiding implies computational context-hiding.

Remark 2.20 (Other Properties for Homomorphic Authenticators). Other properties for HomAuth schemes including compactness and composability have been considered in previous works [GW13, CFGN14, GVW15b, CF18]. Although, compactness is usually regarded as one of the standard property of HomAuth schemes, for our particular application of constructing NIZKs, neither compactness nor composability are necessarily. Nonetheless, it may be more appealing to use a compact HomAuth scheme since it allows us to construct PP-NIZKs with a *succinct* proof size. We note that our HomAuth scheme in Section 4 will satisfy compactness.

Kim and Wu proved that context-hiding HomAuth schemes can be used to construct PP-NIZKs. (See Appendix C for details.

Theorem 2.21 ([KW18a], Theorem 4.4). *If there exists a secret key encryption scheme whose decryption algorithm is computable in $\mathbf{NC}^1$ and an HomAuth scheme for $\mathbf{NC}^1$ with computational (resp. statistical) unforgeability and computational context-hiding, then there exists PP-NIZK with computational (resp. statistical) soundness and non-programmable CRS zero-knowledge. Moreover, if the underlying HomAuth scheme is publicly verifiable (i.e., HomSig scheme), then we can make the resulting PP-NIZK publicly verifiable (i.e., DP-NIZK) by weakening the zero-knowledgeness to programmable CRS zero-knowledge.*

3 DV-NIZK from CDH via FLS Transform

In this section, we construct a DV-NIZK from the CDH assumption over pairing-free groups based on the FLS construction [FLS99] for CRS-NIZKs from TDPs. More formally, we prove the following theorem.

Theorem 3.1. *If the CDH assumption holds on a pairing-free group, then there exists an (adaptive multi-theorem) DV-NIZK proof system for all **NP** languages.*

If we only require *non-adaptive* zero-knowledge where an adversary has to declare statements on which he sees proofs before seeing a common reference string and verification key, then the simple construction described in the introduction works. (More detailed description of the non-adaptive scheme can be found in Appendix A.) On the other hand, for proving adaptive zero-knowledge, we need more efforts. Namely, we construct a DV-NIZK with adaptive zero-knowledge in the following steps:

1. We first construct a variant of DV-NIZK proof system (which we call the *base proof system*) with a special syntax satisfying a relaxed notion of soundness and adaptive *single-theorem* zero-knowledge. We construct it from a NIZK proof system in the hidden-bits model based on the CDH assumption over pairing-free groups. This is done by applying the FLS construction [FLS99] along with the twin-DH technique. A relaxed notion of adaptive zero-knowledge is achieved by using a technique often used in non-committing encryption.

2. We then construct an adaptive *designated-verifier non-interactive witness indistinguishable* (DV-NIWI) proof for all **NP** languages by running many copies of the base proof system in parallel.

3. Finally, we transform our adaptive DV-NIWI proofs into adaptive multi-theorem DV-NIZK proofs by using pseudorandom generators via the transformation of Feige, Lapidot, and Shamir [FLS99] (i.e., the technique of FLS is applicable to the DV-NIZK setting).

3.1 Preliminaries

We introduce the Goldreich-Levin hardcore function $\mathsf{GL}(a;r)$. This is defined by $\mathsf{GL}(a;r) := \langle a, r\rangle := \bigoplus_{j=1}^{u}(a_j \cdot r_j)$ where $a, r \in \{0,1\}^u$ and σ_j denotes the j-th bit of a string σ. In fact, we use groups in our construction and the input to GL is an element in $\mathbb{G}$. Thus, we interpret a group element $g^{r_i} \in \mathbb{G}$ as a u-bit-string.

Theorem 3.2 (Goldreich-Levin Theorem (adapted) [GL89]). *Assuming that the CDH assumption holds, it holds that*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{GL\text{-}cdh}}_{\mathcal{A}}(\kappa,0)=1] - \Pr[\mathsf{Expt}^{\mathsf{GL\text{-}cdh}}_{\mathcal{A}}(\kappa,1)=1]\right| \leq \mathsf{negl}(\kappa),$$

where the experiment $\mathsf{Expt}^{\mathsf{GL\text{-}cdh}}_{\mathcal{A}}(\kappa,\mathsf{coin})$ is defined as follows.

$\underline{\mathsf{Expt}^{\mathsf{GL\text{-}cdh}}_{\mathcal{A}}(\kappa,\mathsf{coin})}$
Samples $(\mathbb{G},p,g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $R \xleftarrow{\$} \{0,1\}^u$, *and* $x,y \xleftarrow{\$} \mathbb{Z}_p$.
If $\mathsf{coin}=1$, *then* $\rho \xleftarrow{\$} \{0,1\}$, *else if* $\mathsf{coin}=0$, *then* $\rho := \mathsf{GL}(g^{xy};R)$.
Output $\mathsf{coin}' \leftarrow \mathcal{A}(1^\kappa,\mathbb{G},p,g,g^x,g^y,R,\rho)$

Next, we introduce a theorem called twin-DH trapdoor test which enables one to check if a tuple (g, X, Y, Z) is a DH-tuple without knowing the discrete logarithm of X or Y by using a special trapdoor.

Theorem 3.3 (Twin-DH Trapdoor Test [CKS09]). *For any $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(\kappa)$ and function F, it holds that*

$$\Pr\left[(Z^{\alpha}\widehat{Z} \stackrel{?}{=} Y^{\beta}) \neq ((Z \stackrel{?}{=} Y^{x}) \wedge (\widehat{Z} \stackrel{?}{=} Y^{\hat{x}})) \;\middle|\; \begin{array}{rl} X & \stackrel{\$}{\leftarrow} \mathbb{G}, \\ \alpha, \beta & \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ \widehat{X} := g^{\beta}/X^{\alpha}, \\ (Y, Z, \widehat{Z}) & \leftarrow F((\mathbb{G}, p, g), X, \widehat{X}) \end{array}\right] \leq 1/p,$$

where $X = g^{x}$ and $\widehat{X} = g^{\hat{x}}$.

We introduce the notion of witness indistinguishability.

Definition 3.4 (Adaptive WI (in the DV model)). *We say that a proof system Π satisfies adaptive witness indistinguishability if for all PPT adversaries $\mathcal{A}$ that makes arbitrary number of queries (resp. at most 1 query), if we run $(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(1^{\kappa})$, then we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs},\cdot,\cdot,\cdot)}(1^{\kappa}, \mathsf{crs}, k_{\mathsf{V}}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\mathsf{crs},\cdot,\cdot,\cdot)}(1^{\kappa}, \mathsf{crs}, k_{\mathsf{V}}) = 1]\right| = \mathsf{negl}(\kappa),$$

where $\mathcal{O}_b(\mathsf{crs}, x, w_0, w_1)$ outputs $\mathsf{Prove}(\mathsf{crs}, x, w_b)$ if $(x, w_0) \in \mathcal{R} \wedge (x, w_1) \in \mathcal{R}$ and $\perp$ otherwise.

Definition 3.5 (Adaptive NIWI). *We say that a proof system Π is adaptive designated-verifier non-interactive witness indistinguishable proof system if Π satisfies completeness, soundness in Definition 2.7 (in the designated-verifier model), and adaptive witness indistinguishability in Definition 3.4.*

We then formally define a NIZK proof in the hidden-bits model, which will be used as a building block in our construction.

Definition 3.6. *A NIZK proof in the hidden-bits model (HBM) for $\mathcal{L}$ is defined by the following two polynomial time algorithms:*

$\mathsf{Prove}(1^{\kappa}, x, w, \rho) \rightarrow (\pi, I)$*: The prover's algorithm takes as input the security parameter 1^{κ}, a statement x, a witness w, and a hidden random string $\rho \in \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}$, and outputs a proof π and a set of indices $I \subseteq [\ell_{\mathsf{hrs}}(\kappa)]$ where $\ell_{\mathsf{hrs}}(\cdot)$ is a polynomial of κ.*

$\mathsf{Verify}(1^{\kappa}, x, \pi, I, \rho_{|I}) \rightarrow \top$ *or* $\perp$*: The verifier's algorithm takes as input the security parameter, a statement x, a proof π, an index set I, a substring $\rho_{|I} := \{\rho_i\}_{i \in I}$, where ρ_i is the i-th bit of ρ, and outputs $\top$ to indicate acceptance of the proof and $\perp$ otherwise.*

Completeness. *For all $x \in \mathcal{L}$ and w such that $(x, w) \in \mathcal{R}$, we have*

$$\Pr[\rho \stackrel{\$}{\leftarrow} \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}, (\pi, I) \leftarrow \mathsf{Prove}(1^{\kappa}, x, w, \rho) : \mathsf{Verify}(1^{\kappa}, x, \pi, I, \rho_{|I}) = \top] = 1.$$

Soundness. *For all (possibly inefficient) adversaries $\mathcal{A}$, we have*

$$\epsilon_{\mathsf{HBM}} := \Pr[\rho \stackrel{\$}{\leftarrow} \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}, (x, \pi, I) \leftarrow \mathcal{A}(1^{\kappa}, \rho) : x \notin \mathcal{L} \wedge \mathsf{Verify}(1^{\kappa}, x, \pi, I, \rho_{|I}) = \top] = \mathsf{negl}(\kappa).$$

We call ϵ_{HBM} soundness error.

Zero-Knowledge. *There exists a PPT simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\begin{aligned}\Big|\Pr[(x, w) \leftarrow \mathcal{A}_1(1^{\kappa}), \rho \stackrel{\$}{\leftarrow} \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}, (\pi, I) \leftarrow \mathsf{Prove}(1^{\kappa}, x, w, \rho) : \mathcal{A}_2(x, \pi, I, \rho_{|I}) = 1] \\ - \Pr[(x, w) \leftarrow \mathcal{A}_1(1^{\kappa}), (\pi, I, \rho_{|I}) \leftarrow \mathcal{S}(1^{\kappa}, x) : \mathcal{A}_2(x, \pi, I, \rho_{|I}) = 1]\Big| = \mathsf{negl}(\kappa).\end{aligned}$$

Theorem 3.7 (NIZK for all NP languages in the HBM [FLS99]). *Unconditionally, there exists NIZK proof systems for all **NP** languages in the HBM with soundness error $\epsilon_{\mathsf{HBM}} \leq 2^{-cn\kappa}$ where $c > 1$ is a constant, n is polynomially related to the size of the circuit computing the **NP** language, κ is the security parameter, and $\ell_{\mathsf{hrs}} = \mathsf{poly}(\kappa, n)$.*

3.2 Constructing DV-NIWI

The goal of this subsection is proving the following theorem.

Theorem 3.8. *Assume that the CDH assumption over paring-free group holds, then there exists an adaptive DV-NIWI for all* ***NP*** *languages.*

Here, we sketch our high-level construction. First, we present our so-called base proof system bP, and then convert it into an adaptive DV-NIWI proof system. Here, the base proof system bP is *not* a standard DV-NIWI proof system since it has a slightly different syntax. Namely, the proving and verification algorithms of the base proof system take an auxiliary string s as input in addition to (crs, x, w) and $(\mathsf{crs}, k_{\mathsf{V}}, x, \pi)$, respectively. We show that the base proof system satisfies two properties called *relaxed* soundness, which means that an adversary cannot forge a proof *if s is fixed*, and *relaxed* zero-knowledge, which means that a proof can be simulated without a witness *if s is randomly chosen*. Observe that if we were to convert the prover to sample s on its own and include it in the proof, then the syntax fits that of DV-NIWI. However, such a simple conversion of our base proof system bP into a DV-NIWI will not work as the acquired DV-NIWI will not have soundness. Namely, the relaxed soundness of bP does not prevent a cheating prover from forging a proof if he is allowed to choose s himself. To resolve this problem, we use a similar idea used by Dwork and Naor [DN07]. Our construction of an adaptive DV-NIWI proof system consists of running many copies of the base proof system using a single common auxiliary input s for all copies. Then, when the number of copies is sufficiently large, soundness of the scheme can be proven from the union bound on all possible s. Moreover, since the relaxed zero-knowledge implies witness indistinguishability, and witness indistinguishability is preserved under parallel repetitions, we can prove the witness indistinguishability of our DV-NIWI.

Base proof system. First, we introduce the syntax and security properties of the base proof system bP. Note that bP is merely an intermediate system introduced for a modular exposition and not a standard NIZK proof system.

Definition 3.9 (Syntax of base proof system). *A base proof system bP consists of the following three polynomial time algorithms.*

$\mathsf{bP.Setup}(1^\kappa) \to (\mathsf{crs}, k_{\mathsf{V}})$*: The setup algorithm takes as input the security parameter 1^κ and outputs a common reference string crs, and a verification key k_{V}.*

$\mathsf{bP.Prove}(\mathsf{crs}, x, w, s) \to \pi$*: The prover's algorithm takes as input a common reference string crs, a statement x, a witness w, and a fixed string $s \in \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}$, and outputs a proof π.*

$\mathsf{bP.Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi, s) \to \top$ *or* $\bot$*: The verifier's algorithm takes as input a common reference string crs, a verification key k_{V}, a statement x, a proof π, and a fixed string $s \in \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}$, and outputs $\top$ to indicate acceptance of the proof and $\bot$ otherwise.*

Definition 3.10 (Security of base proof system). *A base proof system is required to satisfy the following three properties.*

Correctness: *For all pairs $(x, w) \in \mathcal{R}$ and $s \in \{0,1\}^{\ell_{\mathsf{hrs}}(\kappa)}$, if we run $(\mathsf{crs}, k_{\mathsf{V}}) \xleftarrow{\$} \mathsf{bP.Setup}(1^\kappa)$, then we have*

$$\Pr[\pi \xleftarrow{\$} \mathsf{bP.Prove}(\mathsf{crs}, x, w, s) : \mathsf{bP.Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi, s) = \top] = 1$$

Relaxed ϵ-soundness: *For any fixed $s \in \{0,1\}^{\ell_{\mathsf{hrs}}}$, it holds that all (possibly inefficient) adversaries $\mathcal{A}$,*

$$\Pr[\mathsf{Expt}^{\mathsf{r\text{-}snd}}_{\mathcal{A}}(1^\kappa, s) = \top] < \epsilon,$$

where ϵ is the soundness error of bP, and the experiment $\mathsf{Expt}^{\mathsf{r\text{-}snd}}_{\mathcal{A}}(1^\kappa)$ is defined as follows.

$\mathsf{Expt}^{\mathsf{r\text{-}snd}}_{\mathcal{A}}(1^\kappa, s)$
$(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$,
$(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{bP.Verify}(\mathsf{crs}, k_{\mathsf{V}}, \cdot, \cdot, s)}(1^\kappa, \mathsf{crs}, s)$,
If $x^* \notin \mathcal{L} \wedge \mathsf{bP.Verify}(\mathsf{crs}, k_{\mathsf{V}}, x^*, \pi^*, s) = \top$*, then outputs* 1,
Otherwise, outputs 0.

This is basically the same as the standard soundness except that $\mathcal{A}$ must use a fixed s.

Relaxed zero-knowledge: *There exists a PPT simulation algorithm* $\mathsf{bP}.\mathcal{S} = (\mathsf{bP}.\mathcal{S}_1, \mathsf{bP}.\mathcal{S}_2)$ *that satisfies the following. For all (stateful) PPT adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{r\text{-}real}}_{\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{r\text{-}sim}}_{\mathcal{A},\mathcal{S}}(1^\kappa) = 1]\right| = \mathsf{negl}(\kappa),$$

where experiments $\mathsf{Expt}^{\mathsf{r\text{-}real}}_{\mathcal{A}}$ *and* $\mathsf{Expt}^{\mathsf{r\text{-}sim}}_{\mathcal{A},\mathcal{S}}$ *are defined as follows.*

$\underline{\mathsf{Expt}^{\mathsf{r\text{-}real}}_{\mathcal{A}}}$
$(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$,
$(x, w) \leftarrow \mathcal{A}(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$,
$s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$,
If $(x, w) \in \mathcal{R}$, $\pi \leftarrow \mathsf{bP.Prove}(\mathsf{crs}, x, w, s)$,
otherwise $\pi := \bot$,
$b' \leftarrow \mathcal{A}(\pi, s)$
outputs b'

$\underline{\mathsf{Expt}^{\mathsf{r\text{-}sim}}_{\mathcal{A},\mathcal{S}}}$
$(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}}) \leftarrow \mathsf{bP}.\mathcal{S}_1(1^\kappa)$,
$(x, w) \leftarrow \mathcal{A}(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$,

If $(x, w) \in \mathcal{R}$, $(\pi, s) \leftarrow \mathsf{bP}.\mathcal{S}_2(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}}, x)$,
otherwise $\pi := \bot$,
$b' \leftarrow \mathcal{A}(\pi, s)$
outputs b'

We present a base proof system $\mathsf{bP} := (\mathsf{bP.Setup}, \mathsf{bP.Prove}, \mathsf{bP.Verify})$ based on a NIZK proof system in the HBM $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$ (with hidden-random-string-length $\ell_{\mathsf{hrs}}(\kappa)$) and the CDH assumption. Note that we use the $\mathsf{GGen}(1^\kappa)$ algorithm to generate $(\mathbb{G}, p, g)$ where $2^{2\kappa} \leq p$ throughout Section 3. Hereafter, we simply write ℓ_{hrs} instead of $\ell_{\mathsf{hrs}}(\kappa)$ for ease of notation.

$\mathsf{bP.Setup}(1^\kappa)$**:** This algorithm generates the following parameters.

1. Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
2. Samples $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}]$ and $b \in \{0,1\}$ and a common reference string $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}^{2\ell_{\mathsf{hrs}}}$ uniformly at random.
3. Sets $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
4. Samples $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$.
5. Outputs a common reference string $\mathsf{crs} := (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.

We can interpret crs as $(\{X_{i,b}, \widehat{X}_{i,b}, R_i\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}) \in \mathbb{G}^{4\ell_{\mathsf{hrs}}} \times \{0,1\}^{\ell_{\mathsf{hrs}}u}$, where u is the length of the binary representation of a group element.

$\mathsf{bP.Prove}(\mathsf{crs}, x, w, s)$**:** This algorithm does the following.

1. Parses $\mathsf{crs} = (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ where $\overline{\mathsf{crs}} = \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} = \{\widehat{X}_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\overline{R} = \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, and $s \in \{0,1\}^{\ell_{\mathsf{hrs}}}$.
2. Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
3. Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ and $\rho_i = \mathsf{GL}(Z_i; R_i)$ for $i \in [\ell_{\mathsf{hrs}}]$.
4. Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$ where $\rho := \rho_1\| \cdots \|\rho_{\ell_{\mathsf{hrs}}}$.
5. Outputs a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.

$\mathsf{bP.Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi, s)$**:** This algorithm parses $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_{,i})\}_{i\in I}, T)$, $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\mathsf{crs} = (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ where $\overline{\mathsf{crs}} = \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} = \{\widehat{X}_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\overline{R} = \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, and $s \in \{0,1\}^{\ell_{\mathsf{hrs}}}$. This algorithm does the following.

- For all $i \in I$,
  1. Verifies that $\mathsf{Test}_{\mathsf{TDH}}((\alpha_{i,s_i}, \beta_{i,s_i}), X_{i,s_i}, \widehat{X}_{i,s_i}, T, Z_i, \widehat{Z}_i) = \top$, where $\mathsf{Test}_{\mathsf{TDH}}$ is defined in Figure 1. If this equation does not hold, then the verification algorithm immediately outputs $\bot$.

The trapdoor test $\mathsf{Test}_{\mathsf{TDH}}((\alpha,\beta),X,\widehat{X},Y,Z,\widehat{Z})$

1. Verifies that $Z^{\alpha}\cdot\widehat{Z}=Y^{\beta}$. If it holds, then outputs $\top$, else $\bot$.

Figure 1: The algorithm $\mathsf{Test}_{\mathsf{TDH}}((\alpha,\beta),X,\widehat{X},Y,Z,\widehat{Z})$ verifies that $Z=Y^{x}$ and $\widehat{Z}=Y^{\widehat{x}}$, that is (g,Y,X,Z) and $(g,Y,\widehat{X},\widehat{Z})$ where $X=g^{x}$ and $\widehat{X}=g^{\widehat{x}}$ are DDH-tuples without $(x,\widehat{x})$.

2. Computes $\rho_i=\mathsf{GL}(Z_i;R_i)$.

- If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^{\kappa},x,\pi_{\mathsf{hbm}},I,\rho_{|I})$.

Unlike the idea outlined in the introduction, the CRS in bP consists of a doubled-line of random elements $(X_{i,0},\widehat{X}_{i,0})$ and $(X_{i,1},\widehat{X}_{i,1})$ for each $i\in[\ell_{\mathsf{hrs}}]$. These doubled-line of random elements are crucial for achieving *adaptive* zero-knowledge. If we only had a singled-line of random elements as the CRS in the introduction, then we would have the following issue: The only way for the ZK-simulator of bP $\mathcal{S}_{\mathsf{bP}}$ to use the ZK-simulator $\mathcal{S}_{\mathsf{hbm}}$ of the NIZK in the HBM, is to feed $\mathcal{S}_{\mathsf{hbm}}$ the statement x output by the adversary. Now, for the simulated proof π, index set I, and hidden bits $\rho_{|I}$ output by $\mathcal{S}_{\mathsf{hbm}}$ to be useful, we must have $\rho_i=\mathsf{GL}(X_i^{\tau};R_i)$ for all $i\in I$ where τ is some element simulated by $\mathcal{S}_{\mathsf{bP}}$. However, due to soundness, if the CRS was only a single-line of random elements $(X_i,\widehat{X}_i)$, then there exists no τ with overwhelming probability such that the above condition holds. Therefore, $\mathcal{S}_{\mathsf{bP}}$ must choose τ and program the singled-line of random elements $(X_i,\widehat{X}_i)$ in the CRS conditioned on $\rho_i=\mathsf{GL}(X_i^{\tau};R_i)$ for all $i\in I$ in order to appropriately use $\mathcal{S}_{\mathsf{hbm}}$. However, since ρ_i is only output as the result of feeding $\mathcal{S}_{\mathsf{hbm}}$ with the statement x, $\mathcal{S}_{\mathsf{bP}}$ can only set the CRS *after* it is given the statement x from the adversary. To overcome this problem, we use the technique of non-committing encryption. Namely, we let CRS be a doubled-line of random elements $(X_{i,0},\widehat{X}_{i,0})$ and $(X_{i,1},\widehat{X}_{i,1})$. In the real-scheme the fixed string $s\in\{0,1\}^{\ell_{\mathsf{hrs}}}$ dictates which ℓ_{hrs}-random elements $(X_{i,s_i},\widehat{X}_{i,s_i})_{i\in[\ell_{\mathsf{hrs}}]}$ a prover must use. Then during the adaptive ZK proof, $\mathcal{S}_{\mathsf{bP}}$ will prepare the CRS so that $\{\mathsf{GL}(X_{i,0}^{\tau};R_i),\mathsf{GL}(X_{i,1}^{\tau};R_i)\}=\{0,1\}$ *without* seeing the statement x. Then after the adversary outputs the statement x, it runs $\mathcal{S}_{\mathsf{hbm}}$, and samples a string s so that $\rho_i=\mathsf{GL}(X_{i,s_i}^{\tau};R_i)$ for all $i\in I$.

Before we prove the security of bP, we introduce several lemmas that are useful for proving the security of bP.

Lemma 3.11. *The distribution of* $(\mathsf{GL}(X;R),R)$ *is statistically indistinguishable from uniform. That is, it holds that*

$$\mathsf{Real}(\kappa)\overset{\mathrm{stat}}{\approx}\mathsf{Ideal}(\kappa),$$

where the experiments Real *and* Ideal *are defined as follows.*

$\underline{\mathsf{Real}(\kappa)}$	$\underline{\mathsf{Ideal}(\kappa)}$
$(\mathbb{G},p,g)\xleftarrow{\$}\mathsf{GGen}(1^{\kappa})$	$(\mathbb{G},p,g)\xleftarrow{\$}\mathsf{GGen}(1^{\kappa})$
$R\xleftarrow{\$}\{0,1\}^{u}$,	$R\xleftarrow{\$}\{0,1\}^{u}$,
$X\xleftarrow{\$}\mathbb{G}$,	
$b:=\mathsf{GL}(X;R)$,	$b\xleftarrow{\$}\{0,1\}$,
Output (b,R)	*Output* (b,R)

Proof. First, $\mathsf{GL}(X;R)$ is a universal hash since it is the inner-product function. Therefore, we can apply the leftover hash lemma, and obtain $\Delta((\mathsf{GL}(X;R),R),(b,R))\leq 2^{-\kappa-1/2}$ since we set $m=1$ and $2^{2\kappa}\leq p$ and it holds that $H_{\infty}(X)\geq 2\kappa$. □

In the proofs in this section, we frequently use the sampling algorithm in the following lemma.

Lemma 3.12. *There exists a PPT algorithm that takes as inputs uniformly random* $\rho\xleftarrow{\$}\{0,1\}$, $R\xleftarrow{\$}\{0,1\}^{u}$, *and* $T\xleftarrow{\$}\mathbb{G}$, *and outputs random* $\chi\xleftarrow{\$}\mathbb{Z}_p$ *such that* $\mathsf{GL}(T^{\chi};R)=\rho$ *with probability* $1-\mathsf{negl}(\kappa)$. *Furthermore, sampling* ρ *and* χ *in such a way they have the same joint distribution as first sampling* $\chi\xleftarrow{\$}\mathbb{Z}_p$ *and then setting* $\rho=\mathsf{GL}(T^{\chi};R)$ *for all but a negligible fraction of fixed* R *and* T.

Proof. By Lemma 3.11, for all but negligible fraction of $R \in \{0,1\}^u$, when we sample $\chi \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{GL}(T^\chi; R)$ is statistically indistinguishable from a uniformly random bit. Therefore, we can sample random χ such that $\mathsf{GL}(T^\chi; R) = \rho$ except with negligible probability. □

Lemma 3.13. *We define the following experiments* $\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, b)$ *between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$*, chooses* $x \xleftarrow{\$} \mathbb{Z}_p$ *and* $R_i \xleftarrow{\$} \{0,1\}^u$ *for all* $i \in [\ell]$*, and sends* $(1^\kappa, (\mathbb{G}, p, g), g^x, R_1 \| \cdots \| R_\ell)$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *chooses* $\rho^* \in \{0,1\}^\ell$ *and sends* ρ^* *to the challenger.*
3. *If* $b = 0$*, the challenger chooses* $y_i \xleftarrow{\$} \mathbb{Z}_p$ *uniformly at random for* $i \in [\ell]$ *(ignoring* ρ^**), and sends* $(g^{y_1}, \ldots, g^{y_\ell})$ *to* $\mathcal{A}$*. Otherwise, the challenger chooses* $y_i \xleftarrow{\$} \mathbb{Z}_p$ *such that* $\mathsf{GL}(g^{xy_i}; R_i) = \rho^*_i$*, and sends* $(g^{y_1}, \ldots, g^{y_\ell})$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* $b' \in \{0,1\}$*. The experiment outputs* b'*.*

If the CDH assumption holds, then for any PPT $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 1) = 1]\right| \leq \mathsf{negl}(\kappa).$$

Proof of Lemma 3.13. We will prove that the lemma holds for $\ell = 1$ if the CDH assumption holds. This immediately implies Lemma 3.13 due to the standard hybrid argument.

We consider hybrid games $\mathsf{Hyb}_1, \mathsf{Hyb}_2, \mathsf{Hyb}_3$ as follows.

$\underline{\mathsf{Hyb}_1}$
$(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$
$x \xleftarrow{\$} \mathbb{Z}_p$, $y \xleftarrow{\$} \mathbb{Z}_p$, $R \xleftarrow{\$} \{0,1\}^u$,
$\rho \xleftarrow{\$} \{0,1\}$,
$\rho^* \leftarrow \mathcal{A}(1^\kappa, (\mathbb{G}, p, g), g^x, R)$,

if $\rho^* = \rho$, $b' \leftarrow \mathcal{A}(g^y)$
else aborts,
outputs b'

$\underline{\mathsf{Hyb}_2}$
$(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$
$x \xleftarrow{\$} \mathbb{Z}_p$, $y \xleftarrow{\$} \mathbb{Z}_p$, $R \xleftarrow{\$} \{0,1\}^u$,
$\rho \leftarrow \mathsf{GL}(g^{xy}; R)$,
$\rho^* \leftarrow \mathcal{A}(1^\kappa, (\mathbb{G}, p, g), g^x, R)$,
$y' \xleftarrow{\$} \mathbb{Z}_p$ s.t. $\mathsf{GL}(g^{xy'}; R) = \rho^*$,
if $\rho^* = \rho$, $b' \leftarrow \mathcal{A}(g^{y'})$
else aborts,
outputs b'

$\underline{\mathsf{Hyb}_3}$
$(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$
$x \xleftarrow{\$} \mathbb{Z}_p$,$R \xleftarrow{\$} \{0,1\}^u$
$\rho \xleftarrow{\$} \{0,1\}$,
$\rho^* \leftarrow \mathcal{A}(1^\kappa, (\mathbb{G}, p, g), g^x, R)$,
$y' \xleftarrow{\$} \mathbb{Z}_p$ s.t. $\mathsf{GL}(g^{xy'}; R) = \rho^*$,
if $\rho^* = \rho$, $b' \leftarrow \mathcal{A}(g^{y'})$
else aborts,
outputs b'

It is easy to see $\Pr[\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 0) = 1] = 2\Pr[\mathsf{Hyb}_1(1^\kappa) = 1]$ and $\Pr[\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 1) = 1] = 2\Pr[\mathsf{Hyb}_3(1^\kappa) = 1]$ since $\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 0)$ (resp. $\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{A}}(1^\kappa, 1)$) is the same as Hyb_1 (resp. Hyb_3) as long as $\rho^* = \rho$. Moreover, $\rho^* = \rho$ happens with probability $1/2$ since ρ is uniformly and independently chosen.

We show $\mathsf{Hyb}_1 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_2$ and $\mathsf{Hyb}_2 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_3$ if the CDH assumption holds. We construct an adversary $\mathcal{B}$ of the hardcore problem of the CDH problem.

$\mathsf{Hyb}_1 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_2$: $\mathcal{B}$ is given $(1^\kappa, \mathbb{G}, p, g, g^x, g^y, R, \rho)$ from the challenger of the Goldreich-Levin hardcore game in Theorem 3.2. To use a distinguisher $\mathcal{D}$ of the two games, $\mathcal{B}$ sends $(1^\kappa, (\mathbb{G}, p, g), g^x, R)$ to $\mathcal{D}$ and receives ρ^* from $\mathcal{D}$. If $\rho = \rho^*$, then $\mathcal{B}$ sends g^y to $\mathcal{D}$. $\mathcal{B}$ receives b' from $\mathcal{D}$. If $\rho \neq \rho^*$ $\mathcal{B}$ aborts.

In the case $\rho = \mathsf{GL}(g^{xy}; R)$, then $\mathcal{B}$ simulates Hyb_2 since if $\rho = \rho^*$, then the joint distribution of y and $\rho^* = \mathsf{GL}(g^{xy}; R)$ is the same as that of y' and ρ^* where y' is uniformly chosen from $\mathbb{Z}_p$ subject to $\mathsf{GL}(g^{xy'}; R) = \rho^*$ as in Hyb_2. In the case $\rho \xleftarrow{\$} \{0,1\}$, then $\mathcal{B}$ perfectly simulates Hyb_1. This completes the proof of $\mathsf{Hyb}_1 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_2$.

$\mathsf{Hyb}_2 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_3$: $\mathcal{B}$ is given $(1^\kappa, \mathbb{G}, p, g, g^x, g^y, R, \rho)$ from the challenger of the Goldreich-Levin hardcore game in Theorem 3.2. To use a distinguisher $\mathcal{D}$ of the two games, $\mathcal{B}$ sends $(1^\kappa, (\mathbb{G}, p, g), g^x, R)$ to $\mathcal{D}$ and receives ρ^* from $\mathcal{D}$. If $\rho = \rho^*$, then $\mathcal{B}$ chooses $y' \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}(g^{xy'}; R) = \rho^*$ and sends $g^{y'}$ to $\mathcal{D}$. $\mathcal{B}$ can find such a y' by the sampling algorithm in Lemma 3.12. $\mathcal{B}$ receives b' from $\mathcal{D}$. If $\rho \neq \rho^*$, $\mathcal{B}$ aborts.

In the case $\rho = \mathsf{GL}(g^{xy}; R)$, then $\mathcal{B}$ perfectly simulates Hyb_2. In the case $\rho \xleftarrow{\$} \{0,1\}$, then $\mathcal{B}$ perfectly simulates Hyb_3. This completes the proof of $\mathsf{Hyb}_2 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_3$.

This completes the proof of Lemma 3.13. □

We also prepare a corollary, which immediately follows from Lemma 3.13.

Corollary 3.14. *We define the following experiments* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{left\text{-}right}}(1^\kappa, b)$ *between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$*, chooses* $x \xleftarrow{\$} \mathbb{Z}_p$ *and* $R \xleftarrow{\$} \{0,1\}^u$*,* $y_d \xleftarrow{\$} \mathbb{Z}_p$ *such that* $\mathsf{GL}(g^{xy_d}; R) = d \oplus b$ *for* $d \in \{0,1\}$*, and sends* $(1^\kappa, (\mathbb{G}, p, g), g^x, g^{y_0}, g^{y_1})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs a guess* $b' \in \{0,1\}$*. The experiment outputs* b'*.*

If the CDH assumption holds, then for any PPT $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{left\text{-}right}}(1^\kappa, 0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{left\text{-}right}}(1^\kappa, 1) = 1]\right| \leq \mathsf{negl}(\kappa).$$

Security of bP**.** Now, we prove the security of the base proof system.

Lemma 3.15 (Correctness). *Our base proof system* bP *satisfies the correctness in Definition 3.10.*

Proof of Lemma 3.15. If $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ is an honestly generated proof, then we have $Z_i = X_{i,s_i}^\tau$ and $\widehat{Z}_i = \widehat{X}_{i,s_i}^\tau$ for $i \in I$. Therefore, an honestly generated proof passes the twin-DH trapdoor test $\mathsf{Test}_{\mathsf{TDH}}$ and a verifier obtains the valid hidden bits ρ via $\mathsf{GL}(Z_i; R_i)$. Thus, we can use the correctness of HBM and the correctness of bP follows. □

Lemma 3.16 (Relaxed Soundness). *If the soundness error of* HBM *is* ϵ_{HBM} *and the number of verification queries is at most* q_v*, then* bP *satisfies the relaxed* $(p \cdot \epsilon_{\mathsf{HBM}} + (q_v + 1)/p)$*-soundness defined in Definition 3.10.*

Proof of Lemma 3.16. We define a sequence of hybrid games. In the following games, $s \in \{0,1\}^{\ell_{\mathsf{hrs}}}$ is fixed.

Game_0**:** This game is the original experiment defined in Lemma 3.16. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Samples $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}], b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}^{2\ell_{\mathsf{hrs}}}$.
   - Samples $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b} := X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}], b\in\{0,1\}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}], b\in\{0,1\}}$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, s)$.
3. The experiment plays the role of the verification oracle when $\mathcal{A}$ sends (x', π') as a verification query or the final output as follows.
   - Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i\in I'}, T')$.
   - For all $i \in I'$
     (a) Verifies that $\mathsf{Test}_{\mathsf{TDH}}((\alpha_{i,s_i}, \beta_{i,s_i}), T', Z'_i, \widehat{Z}'_i) = 1$. If this equation does not hold, then the verification oracle immediately returns $\perp$.
     (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
   - If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.
4. $\mathcal{A}$ outputs (x^*, π^*). If $x^* \notin \mathcal{L}$ and π^* passes the verification, then the game outputs 1.

Game_1**:** This game is the same as Game_0 except that it changes how to generate crs and verify proofs as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Samples $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\overline{\mathsf{crs}} := \{X_{i,b} := g^{\chi_{i,b}}\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$.
- Samples $\hat{\chi}_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b} := g^{\hat{\chi}_{i,b}}\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$.

2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, s)$.
3. When $\mathcal{A}$ sends (x', π') as a verification query or the final output, the experiment does the following.
   - Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i \in I'}, T')$.
   - For all $i \in I'$
     (a) Verifies that $Z'_i = (T')^{\chi_{i,s_i}}$ and $\widehat{Z}'_i = (T')^{\hat{\chi}_{i,s_i}}$ for all $i \in [\ell_{\mathsf{hrs}}]$. If this equation does not hold, then the verification oracle immediately returns $\perp$.
     (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
   - If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.
4. $\mathcal{A}$ outputs (x^*, π^*). If $x^* \notin \mathcal{L}$ and π^* passes the modified verification procedure defined above, then the game outputs 1.

Game_2**:** This game is the same as Game_1 except that the game guesses $\tau \in \mathbb{Z}_p$ at the CRS generation phase and aborts when the guess is wrong. That is, the game first randomly guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$. When $\mathcal{A}$ outputs $(x^*, \pi^* = (\pi^*_{\mathsf{hbm}}, I^*, \{(Z^*_i, \widehat{Z}^*_i)\}_{i \in I'}, T^*)$ as its final output, if $T^* \neq g^\tau$, then this game aborts. Otherwise it works similarly to Game_1.

Game_3**:** This game is the same as Game_2 except that it changes how to generate crs as follows.

1. Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
2. Guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$.
3. Chooses $\rho_i \xleftarrow{\$} \{0,1\}$ for $i \in [\ell_{\mathsf{hrs}}]$.
4. Chooses χ_{i,s_i} and R_i such that $\mathsf{GL}((g^\tau)^{\chi_{i,s_i}}; R_i) = \rho_i$ and sets $X_{i,s_i} := g^{\chi_{i,s_i}}$ for $i \in [\ell_{\mathsf{hrs}}]$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
5. Chooses $\chi_{i,1-s_i} \xleftarrow{\$} \mathbb{Z}_p$ and sets $X_{i,1-s_i} := g^{\chi_{i,1-s_i}}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
6. Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$.
7. Samples $\hat{\chi}_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b} := g^{\hat{\chi}_{i,b}}\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$.
8. Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$.

We will prove indistinguishability of hybrid games below.

Claim 3.17. It holds that $\Pr[\mathsf{Game}_0 = 1] - \Pr[\mathsf{Game}_1 = 1] \leq (q_v + 1)/p$.

Proof. It can be seen that the distribution of $\{X_{i,b}, \widehat{X}_{i,b}\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$ in Game_1 is the same as that in Game_0. We can apply Theorem 3.3 and the claim follows since $\mathcal{A}$ sends at most q_v queries to the verification oracle and the experiment uses the verification oracle once more for deciding if $\mathcal{A}$ succeeds in forging a proof. □

Claim 3.18. It holds that $\Pr[\mathsf{Game}_2 = 1] = \frac{1}{p} \Pr[\mathsf{Game}_1 = 1]$.

Proof. The difference between the two games is that the game guesses $\tau \in \mathbb{Z}_p$ and aborts if the guess is incorrect in Game_2. Since the probability that the guess is correct is $1/p$, the claim follows. □

Claim 3.19. It holds that $\Pr[\mathsf{Game}_2 = 1] = \Pr[\mathsf{Game}_3 = 1]$.

Proof. In Game_2, ρ_i is computed by $\mathsf{GL}(g^{\tau\chi_{i,s_i}}; R_i)$ using the CRS and τ. On the other hand, in Game_3, we sample $\rho_i \xleftarrow{\$} \{0,1\}$. After that, we choose random χ_{i,s_i} and R_i such that $\rho_i = \mathsf{GL}((g^\tau)^{\chi_{i,s_i}}; R_i)$. (We can assume that any group element in $\mathbb{G}$ is not encoded into the all zero string.) These two distributions are identically distributed since $\rho_i = \mathsf{GL}(g^{\tau\chi_{i,s_i}}; R_i)$ is uniformly random over $\{0,1\}$ when $\chi_{i,s_i} \xleftarrow{\$} \mathbb{Z}_p$ and $R_i \xleftarrow{\$} \{0,1\}^u$ are uniformly and randomly chosen. □

Claim 3.20. If the soundness error of HBM is ϵ_{HBM}, then it holds that $\Pr[\mathsf{Game}_3 = 1] \leq \epsilon_{\mathsf{HBM}}$.

Proof. We show that if $\mathcal{A}$ can generate a pair of statement and proof (x^*, π^*) that passes the verification in Game_3, then we can construct a cheating prover $\mathcal{B}$ of $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$. To use $\mathcal{A}$, $\mathcal{B}$ proceeds as follows.

CRS simulation: First, $\mathcal{B}$ is given $(1^\kappa, \rho)$. $\mathcal{B}$ simulates crs as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
- Guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$.
- Chooses χ_{i,s_i} and R_i such that $\mathsf{GL}((g^\tau)^{\chi_{i,s_i}}; R_i) = \rho_i$ and sets $X_{i,s_i} := g^{\chi_{i,s_i}}$ for $i \in [\ell_{\mathsf{hrs}}]$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Chooses $\chi_{i,1-s_i} \xleftarrow{\$} \mathbb{Z}_p$ and sets $X_{i,1-s_i} := g^{\chi_{i,1-s_i}}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
- Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
- Samples $\hat{\chi}_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b} := g^{\hat{\chi}_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and gives crs and s to $\mathcal{A}$.

Verification oracle simulation: When $\mathcal{A}$ sends (x', π') as a verification query or the final output, the experiment does the following.

- Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i\in I'}, T')$.
- For all $i \in I'$
  (a) Verifies that $Z'_i = (T')^{\chi_{i,s_i}}$ and $\widehat{Z}'_i = (T')^{\hat{\chi}_{i,s_i}}$ for all $i \in [\ell_{\mathsf{hrs}}]$. If this equation does not hold, then it immediately returns $\perp$.
  (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
- If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.

Forgery: When $\mathcal{A}$ outputs $(x^*, \pi^* = (\pi^*_{\mathsf{hbm}}, I^*, \{(Z^*_i, \widehat{Z}^*_i)\}_{i\in I}, T^*)$, if $T^* \neq g^\tau$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs $(\pi^*_{\mathsf{hbm}}, I^*, \rho_{|I^*})$.

The simulations of $(\mathsf{crs}, k_{\mathsf{V}})$ and the verification oracle are perfect. If Game_3 outputs 1, we have $x^* \notin \mathcal{L}$, $T^* = g^\tau$ and (x^*, π^*) passes the verification oracle in Game_3, which implies we have $Z^*_i = (T^*)^{\chi_{i,s_i}}$ and $\mathsf{HBM.Verify}(1^\kappa, x, \pi^*_{\mathsf{hbm}}, I^*, \rho^*{}_{|I^*}) = 1$ where $\rho^*_i := \mathsf{GL}(Z^*_i; R_i)$. On the other hand, we can see that we have $\mathsf{GL}((g^\tau)^{\chi_{i,s_i}}; R_i) = \rho_i$ by the way of the CRS simulation. Since we have $T^* = g^\tau$, we have $Z^*_i = (g^\tau)^{\chi_{i,s_i}}$ and thus $\rho^*_i = \rho_i$. Hence we have $\mathsf{HBM.Verify}(1^\kappa, x, \pi^*_{\mathsf{hbm}}, I^*, \rho_{|I^*}) = \top$, which means that $\mathcal{B}$ succeeds in forging a proof of HBM. □

By Claims 3.17 to 3.20, we complete the proof of Lemma 3.16. □

Lemma 3.21 (Relaxed ZK). *If the CDH assumption holds with respect to* GGen*, then* bP *satisfies the relaxed ZK defined in Definition 3.10.*

Proof of Lemma 3.21. We fisrt describe the simulator $\mathsf{bP}.\mathcal{S} = (\mathsf{bP}.\mathcal{S}_1, \mathsf{bP}.\mathcal{S}_2)$.

1. $\mathsf{bP}.\mathcal{S}_1(1^\kappa)$ generates $(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}})$ as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
- Chooses $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$, and sets $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
- Chooses $\tau \xleftarrow{\$} \mathbb{Z}_p$, and $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
- Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = b \oplus \widetilde{s}_i$ for $b \in \{0,1\}$. We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.
- Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}} := \{g^{\chi_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} = (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and $\tau_{\mathsf{V}} := (\widetilde{s}, \tau)$.
- Outputs $(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}})$.

2. $\mathsf{bP}.\mathcal{S}_2(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}}, x)$ simulates a proof π and s as follows.
   - Generates $(\pi_{\mathsf{hbm}}, I, \rho_{|I}) \leftarrow \mathsf{HBM}.\mathcal{S}(1^\kappa, x)$.
   - Sets $s_i := \rho_i \oplus \widetilde{s}_i$ for all $i \in I$, samples $s_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$, and sets $s := s_1 \| \cdots \| s_{\ell_{\mathsf{hrs}}}$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Outputs a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ and s.

For proving that proofs simulated by the above simulator is computationally indistinguishable from real ones, we define a sequence of hybrid games. In the following, we assume that the adversary $\mathcal{A}$ only queries (x, w) such that $(x, w) \in \mathcal{R}$ to the proving oracle where $\mathcal{R}$ is the corresponding relation of $\mathcal{L}$. This can be assumed without loss of generality since $\mathcal{A}$ can check if this holds by himself, and if $(x, w) \notin \mathcal{R}$, then the proving oracle just returns $\perp$.

Game_0**:** This game is the real experiment defined in Lemma 3.21. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Samples $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}^{2\ell_{\mathsf{hrs}}}$.
   - Sets $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. Samples $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
4. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Computes $\rho_i := \mathsf{GL}((X_{i,s_i})^\tau; R_i)$ and sets $\rho := \rho_1 \| \cdots \| \rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM}.\mathsf{Prove}(1^\kappa, x, w, \rho)$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
5. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_1**:** This game is the same as Game_0 except that $\rho_{i,b}$ and $X_{i,b}$ are generated in the "reversed" order. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.

- Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
- Chooses $\rho_{i,b} \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}]$ and $b \in \{0,1\}$.
- Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = \rho_{i,b}$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
- Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
- Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1\| \cdots \|R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.

2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. Samples $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
4. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Sets $\rho_i := \rho_{i,s_i}$ and $\rho := \rho_1\| \cdots \|\rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
5. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_2**:** This game is the same as Game_1 except that $\rho_{i,0}$ and $\rho_{i,1}$ are chosen so that they always differ each other. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Samples $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Chooses $\rho_{i,s_i} \xleftarrow{\$} \{0,1\}$ and sets $\rho_{i,1-s_i} := 1 - \rho_{i,s_i}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = \rho_{i,b}$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
   - Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1\| \cdots \|R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Sets $\rho_i := \rho_{i,s_i}$ and $\rho := \rho_1\| \cdots \|\rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_3**:** This game is the same as Game_2 except that generations of ρ and s are delayed until $\mathcal{A}$ outputs (x, w). Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.

- Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
- Samples $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
- Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^{\tau}; R_i) = b \oplus \widetilde{s}_i$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
- Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
- Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.

2. $\mathcal{A}$ is given $(1^{\kappa}, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^{\tau})$ as follows.
   - Samples $\rho \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^{\kappa}, x, w, \rho)$.
   - Sets $s := \rho \oplus \widetilde{s}$.
   - Sets $Z_i := (X_{i,s_i})^{\tau}$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^{\tau}$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^{\tau})$.
4. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_4**:** This game is the same as Game_3 except that s_i is chosen independently of ρ or $\widetilde{s}$ for $i \notin I$. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^{\kappa})$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Samples $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^{\tau}; R_i) = b \oplus \widetilde{s}_i$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
   - Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
2. $\mathcal{A}$ is given $(1^{\kappa}, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^{\tau})$ as follows.
   - Samples $\rho \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^{\kappa}, x, w, \rho)$.
   - Sets $s_i := \rho_i \oplus \widetilde{s}_i$ for all $i \in I$, samples $s_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$, and sets $s := s_1 \| \cdots \| s_{\ell_{\mathsf{hrs}}}$.
   - Sets $Z_i := (X_{i,s_i})^{\tau}$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^{\tau}$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^{\tau})$.
4. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_5**:** This game is the same as Game_4 except that it changes how to generate an NIZK proof in the HBM as follows. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^{\kappa})$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.

- Samples $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
- Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = b \oplus \widetilde{s}_i$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
- Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
- Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1\| \cdots \|R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.

2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ and outputs (x, w).
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Generates $(\pi_{\mathsf{hbm}}, I, \rho_{|I}) \leftarrow \mathsf{HBM}.\mathcal{S}(1^\kappa, x)$.
   - Sets $s_i := \rho_i \oplus \widetilde{s}_i$ for all $i \in I$, samples $s_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$, and sets $s := s_1\| \cdots \|s_{\ell_{\mathsf{hrs}}}$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. (π, s) is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

By the definition, Game_5 is completely the same as the experiment simulated by $\mathsf{bP}.\mathcal{S}$. We will prove the following claims.

Claim 3.22. It holds that $\mathsf{Game}_0 \overset{\mathrm{stat}}{\approx} \mathsf{Game}_1$.

Proof. The difference between two games is how to sample $X_{i,s}$. In Game_0, $X_{i,s}$ is uniformly and randomly chosen. In Game_1, the experiment samples $\rho_i \xleftarrow{\$} \{0,1\}$ and after that $X_{i,b}$ is set to $g^{\chi_{i,b}}$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = \rho_i$. The indistinguishability between these two games directly follows from Lemma 3.12. □

Claim 3.23. If the CDH assumption holds with respect to GGen, then it holds that $\mathsf{Game}_1 \overset{\mathrm{c}}{\approx} \mathsf{Game}_2$.

Proof. If the CDH assumption holds, then we can use Lemma 3.13. Therefore, we show that if $\mathcal{A}$ distinguishes Game_1 and Game_2, then we can construct an adversary $\mathcal{B}$ for the experiment defined in Lemma 3.13.

$\mathcal{B}$ is given $(1^\kappa, (\mathbb{G}, p, g), g^x, R_1\| \cdots \|R_{\ell_{\mathsf{hrs}}})$ and it sets $g^\tau := g^x$. Next, $\mathcal{B}$ chooses $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$ and $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$, and does the following. For $i \in [\ell_{\mathsf{hrs}}]$,

1. Chooses $\rho_{i,s_i} \xleftarrow{\$} \{0,1\}$ and sets $\rho_{i,1-s_i} := 1 - \rho_{i,s_i}$.
2. Chooses $\chi_{i,s_i} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^\tau)^{\chi_{i,s_i}}; R_i) = \rho_{i,s_i}$. We can find such χ_{i,s_i} by the algorithm in Lemma 3.12.

Then, $\mathcal{B}$ sends $(\rho_{1,1-s_1}, \ldots, \rho_{\ell_{\mathsf{hrs}},1-s_{\ell_{\mathsf{hrs}}}})$ to the challenger and receives $(g^{y_1}, \ldots, g^{y_{\ell_{\mathsf{hrs}}}})$. Then, $\mathcal{B}$ sets $X_{i,1-s_i} := g^{y_i}$, $\widehat{X}_{i,1-s_i} := g^{\beta_{i,1-s_i}} \cdot (g^{y_i})^{-\alpha_{i,1-s_i}}$, $X_{i,s_i} := g^{\chi_{i,s_i}}$, and $\widehat{X}_{i,s_i} := g^{\beta_{i,s_i}} \cdot g^{-\alpha_{i,s_i}\chi_{i,s_i}}$ for all $i \in [\ell_{\mathsf{hrs}}]$. $\mathcal{B}$ sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{\widehat{X}_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\overline{R} := R_1\| \cdots \|R_{\ell_{\mathsf{hrs}}}$, and $\mathsf{crs} := \overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$, and gives $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs (x, w). Then $\mathcal{B}$ sets $\rho_i := \rho_{i,s_i}$ and $\rho := \rho_1\| \cdots \|\rho_{\ell_{\mathsf{hrs}}}$, generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM}.\mathsf{Prove}(1^\kappa, x, w, \rho)$, sets $Z_i := (g^\tau)^{r_{i,s_i}}$ and $\widehat{Z}_i := (g^\tau)^{\beta_{i,s_i} - \alpha_i \chi_{i,s_i}}$ for $i \in I$. Then $\mathcal{B}$ sets $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ and gives (π, s) to $\mathcal{A}$. $\mathcal{B}$ outputs as $\mathcal{A}$ outputs.

If g^{y_i} comes from the experiment $\mathsf{Expt}_{\mathcal{B}}^{\mathsf{switch}}(1^\kappa, 0)$, then the simulation above is statistically indistinguishable from Game_1 by Lemma 3.12 since $y_i \xleftarrow{\$} \mathbb{Z}_p$. If g^{y_i} comes from the experiment $\mathsf{Expt}_{\mathcal{B}}^{\mathsf{switch}}(1^\kappa, 1)$, then the simulation above perfectly simulates Game_2 since $y_i \xleftarrow{\$} \mathbb{Z}_p$ are subject to $\mathsf{GL}(g^{xy_i}; R_i) = \rho_{i,1-s_i}$. Therefore, $\mathcal{B}$ can break the CDH assumption by using $\mathcal{A}$ that distinguishes these two hybrid games. This completes the proof of the claim. □

Claim 3.24. It unconditionally holds that $\mathsf{Game}_2 \overset{\mathrm{stat}}{\approx} \mathsf{Game}_3$.

Proof. Suppose that we set $\rho_{i,b} := b \oplus \widetilde{s}_i$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$ by using $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$. This does not change the distribution of $\{\rho_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ since they are uniformly distributed subject to $\rho_{i,0} \neq \rho_{i,1}$. Then $\chi_{i,b}$ is sampled subject to $\mathsf{GL}((g^{\chi_{i,b}})^{\tau}; R_i) = b \oplus \widetilde{s}_i$. Moreover, if we set $\rho_i := \rho_{i,s_i}$ for each $i \in [\ell_{\mathsf{hrs}}]$, we have $\rho_i = s_i \oplus \widetilde{s}_i$, which means $s = \widetilde{s} \oplus \rho$. Since s is uniformly chosen in Game_2, the joint distribution of s and ρ does not change if we first picks $\rho \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$ and then we set $s := \widetilde{s} \oplus \rho$. Now, the game is identical to Game_3. This completes the proof of the claim. □

Claim 3.25. If the CDH assumption holds with respect to GGen, then it holds that $\mathsf{Game}_3 \overset{\mathrm{c}}{\approx} \mathsf{Game}_4$.

Proof. For $k \in \{0, ..., \ell_{\mathsf{hrs}}\}$, we consider hybrids Hyb_k, which is identical to Game_3 except that for all $i \in [k] \setminus I$, s_i is uniformly chosen from $\{0,1\}$ instead of setting $s_i := \rho_i \oplus \widetilde{s}_i$. It is clear that Hyb_0 is identical to Game_3 and $\mathsf{Hyb}_{\ell_{\mathsf{hrs}}}$ is identical to Game_4, Therefore we only have to prove $\mathsf{Hyb}_{k-1} \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_k$ for all $k \in [\ell_{\mathsf{hrs}}]$. To prove this, we introduce an additional hybrid Hyb'_k for $k \in [\ell_{\mathsf{hrs}}]$ which is identical to Hyb_k except that if $k \notin I$, s_k is set to be $1 \oplus (\rho_i \oplus \widetilde{s}_i)$.

We note that if $k \in I$, then Hyb_{k-1}, Hyb_k, and Hyb'_k are completely identical. On the other hand, if $k \notin I$, then Hyb_k is identical to Hyb_{k-1} with probability $1/2$, and identical to Hyb'_k with probability $1/2$ depending on the randomness for choosing s_k. Therefore it suffices to prove $\mathsf{Hyb}_{k-1} \overset{\mathrm{c}}{\approx} \mathsf{Hyb}'_k$ for all $k \in [\ell_{\mathsf{hrs}}]$.

To prove this, suppose that $\mathcal{A}$ distinguishes Hyb_{k-1} and Hyb'_k. Without loss of generality, we can assume that I output by $\mathcal{A}$ satisfies $k \notin I$ since otherwise Hyb_{k-1} and Hyb'_k are completely identical. Then we construct an adversary $\mathcal{B}$ that distinguishes the experiments defined in Corollary 3.14.

1. $\mathcal{B}$ is given $(1^\kappa, (\mathbb{G}, p, g), g^x, g^{y_0}, g^{y_1}, R)$ and generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Chooses $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$, $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus \{k\}$, and sets $R_k := R$.
   - Sets $g^\tau := g^x$.
   - Samples $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - For all $i \in [\ell_{\mathsf{hrs}}] \setminus \{k\}, b \in \{0,1\}$, chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^\tau)^{\chi_{i,b}}; R_i) = b \oplus \widetilde{s}_i$ by the sampling algorithm in Lemma 3.12 and sets $(X_{i,b}, \widehat{X}_{i,b}) := (g^{\chi_{i,b}}, g^{\beta_{i,b} - \alpha_{i,b}\chi_{i,b}})$.
   - Sets $X_{k,b} := g^{y_{b \oplus \widetilde{s_k}}}$ and $\widehat{X}_{k,b} := X_{k,b}^{-\alpha_{k,b}} \cdot g^{\beta_{k,b}}$ for $b \in \{0,1\}$.
   - Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
2. Gives $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs (x, w).
3. $\mathcal{D}$ generates a proof π and s as follows.
   - Samples $\rho \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$.
   - Sets $s_i := \rho_i \oplus \widetilde{s}_i$ for all $i \in \{k, ..., \ell_{\mathsf{hrs}}\} \cup I$, samples $s_i \xleftarrow{\$} \{0,1\}$ for $i \in [k-1] \setminus I$.
   - Sets $Z_i := (g^\tau)^{\chi_{i,s_i}}$, $\widehat{Z}_i := (g^\tau)^{\beta_{i,s_i} - \alpha_{i,s_i}\chi_{i,s_i}}$ for $i \in I$. (Note that we assume $k \notin I$ and thus $\mathcal{B}$ knows $\chi_{i,b}$ for all $i \in I, b \in \{0,1\}$.)
   - Sets $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. Finally, $\mathcal{B}$ gives (π, s) to $\mathcal{A}$, and outputs $\mathcal{A}$'s output as its own output.

If the input of $\mathcal{B}$ comes from $\mathsf{Expt}^{\mathsf{left\text{-}right}}_{\mathcal{A}}(1^\kappa, 0)$ (i.e., y_b is chosen such that $\mathsf{GL}(g^{xy_b}; R) = b$ for $b \in \{0,1\}$), then $\mathcal{B}$ perfectly simulates Hyb_{k-1}, and that comes from $\mathsf{Expt}^{\mathsf{left\text{-}right}}_{\mathcal{A}}(1^\kappa, 1)$ (i.e., y_b is chosen such that $\mathsf{GL}(g^{xy_b}; R) = 1 - b$ for $b \in \{0,1\}$), then $\mathcal{B}$ perfectly simulates Hyb'_k. Therefore if $\mathcal{A}$ distinguishes Hyb_{k-1} and Hyb'_k, then $\mathcal{B}$ distinguishes these two cases, which contradicts Corollary 3.14. □

Claim 3.26. If $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$ is a NIZK proof system in the HBM, then $\mathsf{Game}_4 \overset{\mathrm{stat}}{\approx} \mathsf{Game}_5$.

Proof. We construct a distinguisher $\mathcal{D}$ of HBM by using a distinguisher $\mathcal{A}$ of Game_4 and Game_5. $\mathcal{D}$ is given 1^κ, then does the following.

1. $\mathcal{D}$ generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_{i,b}, \beta_{i,b}) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Samples $\widetilde{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$.
   - Chooses $\chi_{i,b} \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_{i,b}})^\tau; R_i) = b \oplus \widetilde{s}_i$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such $\chi_{i,b}$ by the sampling algorithm in Lemma 3.12.)
   - Sets $X_{i,b} := g^{\chi_{i,b}}$ for all $i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}$.
   - Sets $\overline{\mathsf{crs}} := \{X_{i,b}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$, $\widehat{\mathsf{crs}} := \{X_{i,b}^{-\alpha_{i,b}} \cdot g^{\beta_{i,b}}\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_{i,b}, \beta_{i,b})\}_{i\in[\ell_{\mathsf{hrs}}],b\in\{0,1\}}$.
2. Gives $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs (x, w).
3. Sends (x, w) to the challenger to receive $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$.
4. $\mathcal{D}$ generates a proof π and s as follows.
   - Sets $s_i := \rho_i \oplus \widetilde{s}_i$ for all $i \in I$, samples $s_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$, and sets $s := s_1 \| \cdots \| s_{\ell_{\mathsf{hrs}}}$.
   - Sets $Z_i := (X_{i,s_i})^\tau$ and $\widehat{Z}_i := (\widehat{X}_{i,s_i})^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
5. Gives (π, s) to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.
6. $\mathcal{D}$ outputs b'.

It is clear that $\mathcal{D}$ perfectly simulates Game_4 if $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$ comes from $\mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$, and $\mathcal{D}$ perfectly simulates Game_5 if $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$ comes from $\mathsf{HBM}.\mathcal{S}(1^\kappa, x)$. This completes the proof. □

By Claims 3.22 to 3.26, we complete the proof of Lemma 3.21 □

Construction of DV-NIWI. Here, we present our adaptive DV-NIWI proof system $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ based on the base proof system $\mathsf{bP} := (\mathsf{bP.Setup}, \mathsf{bP.Prove}, \mathsf{bP.Verify})$ that has relaxed ϵ-soundness for some $\epsilon < 1$. We note that we proved that the base proof system satisfies relaxed $(p \cdot \epsilon_{\mathsf{HBM}} + (q_v + 1)/p)$-soundness in Lemma 3.16, and we can make $(p \cdot \epsilon_{\mathsf{HBM}} + (q_v + 1)/p) < 1$ by choosing a parameter for HBM so that $p \cdot \epsilon_{\mathsf{HBM}}$ is negligible. (This is possible by Theorem 3.7). We set an integer ℓ' so that we have $2^{\ell_{\mathsf{hrs}}} \cdot \epsilon^{\ell'} \leq 2^{-\kappa}$. Then Π is described as follows.

$\mathsf{Setup}(1^\kappa)$**:** This algorithm samples $(\mathsf{crs}_j, k_{\mathsf{V}}^{(j)}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$ for $j \in [\ell']$. It sets $\mathsf{crs} := \mathsf{crs}_1 \| \cdots \| \mathsf{crs}_{\ell'}$ and $k_{\mathsf{V}} := k_{\mathsf{V}}^{(1)} \| \cdots \| k_{\mathsf{V}}^{(\ell')}$, and outputs $(\mathsf{crs}, k_{\mathsf{V}})$.

$\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$**:** This algorithm does the following:

1. chooses $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$,
2. generates $\pi_j \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_j, x, w, s)$ for all $j \in [\ell']$,
3. outputs a proof $\pi := (\pi_1, \ldots, \pi_{\ell'}, s)$.

$\mathsf{Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi) \to \top$ **or** $\bot$**:** This algorithm parses $\pi = (\pi_1, \ldots, \pi_{\ell'}, s)$. For all $j \in [\ell']$, it verifies that $\top = \mathsf{bP.Verify}(\mathsf{crs}_j, k_{\mathsf{V}}^{(j)}, x, \pi_j, s)$. If the proof passes all the tests, then this algorithm outputs $\top$, otherwise $\bot$.

Lemma 3.27 (Correctness). Π *satisfies completeness.*

Proof of Lemma 3.27. It is easy to see the completeness holds by Lemma 3.15 since the proof consists of the parallel repetition of the base proof system. □

Lemma 3.28 (Soundness). Π *satisfies soundness.*

Proof of Lemma 3.28. For each fixed s, the probability that an adversary can forge a proof is $\epsilon^{\ell'}$. Since the possible choice for s is $2^{\ell_{\mathsf{hrs}}}$, the probability that an adversary can forge a proof is $2^{\ell_{\mathsf{hrs}}} \cdot \epsilon^{\ell'} \leq 2^{-\kappa}$. □

Lemma 3.29 (WI). *If the CDH assumption holds with respect to* GGen*, then* Π *satisfies adaptive witness-indistinguishability.*

Proof of Lemma 3.29. First, we remark that witness indistinguishability can be reduced to single-theorem witness indistinguishability where an adversary makes only one query by a standard hybrid argument. Therefore in the following, we assume that an adversary makes a query only once.

We define a sequence of hybrid games. Let Hyb_j be a hybrid game where adversaries have oracle access to a hybrid oracle $\mathcal{O}^{(j)}(\mathsf{crs}, \cdot, \cdot, \cdot)$ defined below.

$\mathcal{O}^{(j)}(\mathsf{crs}, x, w_0, w_1)$**:** Chooses $s \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{hrs}}}$, generates $\pi_i^{(1)} \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_1, s)$ for $i \leq j$ and $\pi_i^{(0)} \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_0, s)$ for $i > j$ and returns $\pi := (\pi_1^{(1)}, \ldots, \pi_j^{(1)}, \pi_{j+1}^{(0)}, \ldots, \pi_{\ell'}^{(0)})$.

We can easily see that Hyb_0 and $\mathsf{Hyb}_{\ell'}$ be the transcripts in the adaptive WI experiments where $\mathcal{A}$ has access to $\mathcal{O}_0$ and $\mathcal{O}_1$, respectively. That is, our goal is proving $\mathsf{Hyb}_0 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_{\ell'}$.

To prove it, we will prove $\mathsf{Hyb}_{j-1} \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_j$ for $j = 1, \ldots, \ell'$, which immediately implies $\mathsf{Hyb}_0 \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_{\ell'}$ and completes the proof. We define auxiliary hybrid games $\widetilde{\mathsf{Hyb}}_j$, where the challenger generates $(\mathsf{crs}_i, k_{\mathsf{V}}^{(i)}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$ for $i \in [\ell'] \setminus \{j+1\}$ and $(\mathsf{crs}_j, k_{\mathsf{V}}^{(j+1)}, \tau_{\mathsf{V}}^{(j+1)}) \leftarrow \mathsf{bP}.\mathcal{S}_1(1^\kappa)$ and sets $\mathsf{crs} := \mathsf{crs}_1 \| \cdots \| \mathsf{crs}_{\ell'}$ and $k_{\mathsf{V}} := k_{\mathsf{V}}^{(1)} \| \cdots \| k_{\mathsf{V}}^{(\ell')}$ at the beginning of the game, and an adversary is given $(\mathsf{crs}, k_{\mathsf{V}})$ and has oracle access to $\widetilde{\mathcal{O}}^{(j)}$ defined below.

$\widetilde{\mathcal{O}}^{(j)}(x, w_0, w_1)$**:** Generates $(\pi_{j+1}, s) \leftarrow \mathsf{bP}.\mathcal{S}_2(\tau_{\mathsf{V}}^{(j+1)}, x)$, $\pi_i \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_1, s)$ for $i < j+1$, and $\pi_i \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_0, s)$ for $i > j+1$. Returns $\pi := (\pi_1^{(1)}, \ldots, \pi_j^{(1)}, \pi_{j+1}, \pi_{j+2}^{(0)}, \ldots, \pi_{\ell'}^{(0)})$.

We will prove $\mathsf{Hyb}_{j-1} \overset{\mathrm{c}}{\approx} \widetilde{\mathsf{Hyb}}_{j-1}$ and $\widetilde{\mathsf{Hyb}}_{j-1} \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_j$ in Lemmata 3.30 and 3.31. This completes the proof of Lemma 3.29. □

Lemma 3.30. *If* bP *satisfies the relaxed ZK defined in Definition 3.10, then* $\mathsf{Hyb}_{j-1} \overset{\mathrm{c}}{\approx} \widetilde{\mathsf{Hyb}}_{j-1}$.

Proof of Lemma 3.30. We construct a distinguisher $\mathcal{B}$ for the relaxed zero-knowledge described in Definition 3.10 of the base proof system bP by using a distinguisher $\mathcal{D}$ of Hyb_{j-1} and $\widetilde{\mathsf{Hyb}}_{j-1}$.

First, $\mathcal{B}$ is given $(1^\kappa, \mathsf{crs}^*, k_{\mathsf{V}}^*)$ from the experiment of relaxed zero-knowledge in Definition 3.10. To use $\mathcal{D}$, $\mathcal{B}$ generates $(\mathsf{crs}_i, k_{\mathsf{V}}^{(i)}) \leftarrow \mathsf{bP.Setup}(1^\kappa)$ for $i \in [\ell'] \setminus \{j\}$. $\mathcal{B}$ then sets $\mathsf{crs} := \mathsf{crs}_1 \| \cdots \| \mathsf{crs}_{j-1} \| \mathsf{crs}^* \| \mathsf{crs}_{j+1} \| \cdots \| \mathsf{crs}_{\ell'}$ and $k_{\mathsf{V}} := k_{\mathsf{V}}^{(1)} \| \cdots \| k_{\mathsf{V}}^{(j-1)} \| k_{\mathsf{V}}^* \| k_{\mathsf{V}}^{(j+1)} \| \cdots \| k_{\mathsf{V}}^{(\ell')}$. $\mathcal{B}$ sends $(\mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{D}$ and receives (x, w_0, w_1).

Next, $\mathcal{B}$ simulates a proof as follows. $\mathcal{B}$ sends (x, w_0) to the challenger of the experiment of relaxed zero-knowledge in Definition 3.10, and receives (π^*, s) of bP. Then, $\mathcal{B}$ does the following.

1. For $i < j$, $\mathcal{B}$ generates $\pi_i^{(1)} \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_1, s)$.
2. For $i > j$, $\mathcal{B}$ generates $\pi_i^{(0)} \leftarrow \mathsf{bP.Prove}(\mathsf{crs}_i, x, w_0, s)$.
3. For $i = j$, $\mathcal{B}$ sets $\pi_j := \pi^*$.

$\mathcal{B}$ sends $\pi := (\pi_1^{(1)}, \ldots, \pi_{j-1}^{(1)}, \pi^*, \pi_{j+1}^{(0)}, \ldots, \pi_{\ell'}^{(0)}, s)$ to $\mathcal{D}$. This completes the simulation for $\mathcal{D}$.

If $(\mathsf{crs}^*, k_{\mathsf{V}}^*)$ and (π^*, s) are outputs of $\mathsf{bP.Setup}(1^\kappa)$ and $\mathsf{bP.Prove}(\mathsf{crs}, x, w_0, s)$, then $\mathcal{B}$ perfectly simulates Hyb_{j-1}. If $(\mathsf{crs}^*, k_{\mathsf{V}}^*)$ and (π^*, s) are outputs of $\mathsf{bP}.\mathcal{S}_1(1^\kappa)$ and $\mathsf{bP}.\mathcal{S}_2(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}}, x)$, then $\mathcal{B}$ perfectly simulates $\widetilde{\mathsf{Hyb}}_{j-1}$. Therefore, if $\mathcal{D}$ distinguishes these two hybrid games, then $\mathcal{B}$ can break the zero-knowledge in Lemma 3.21. This complete the proof of Lemma 3.30. □

Lemma 3.31. *If* bP *satisfies the relaxed ZK defined in Definition 3.10, then* $\widetilde{\mathsf{Hyb}}_{j-1} \overset{\mathrm{c}}{\approx} \mathsf{Hyb}_j$.

Proof of Lemma 3.31. We can prove this similarly to Lemma 3.31. □

From Lemmata 3.15, 3.16, 3.21 and 3.27 to 3.29, we obtain Theorem 3.8 presented at the beginning of this sub-section.

3.3 Transformation from DV-NIWI into Multi-Theorem DV-NIZK

In this section, we show how to convert adaptive DV-NIWI into adaptive *multi-theorem* DV-NIZKs for **NP**. This will complete the proof of Theorem 3.1.

The goal of this subsection is proving the following theorem.

Theorem 3.32. *If there exists an adaptive DV-NIWI proof systems and pseudorandom generators, then there exists an adaptive multi-theorem DV-NIZK proof system.*

The transformation is essentially the same as that of Feige et al. [FLS99] (from NIWI to multi-theorem NIZK). However, we consider the *designated-verifier* setting. Thus, we formally prove Theorem 3.32 for confirmation.

We present our adaptive multi-theorem DV-NIZK proof system $\Pi_{\mathsf{zk}} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ based on adaptive DV-NIWI proof system $\Pi_{\mathsf{wi}} := (\mathsf{WI.Setup}, \mathsf{WI.Prove}, \mathsf{WI.Verify})$ for all NP languages. Let $\mathsf{PRG}(\cdot) : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudorandom generator as in Definition 2.4.

$\mathsf{Setup}(1^\kappa)$**:** This algorithm samples $(\mathsf{wi.crs}, \mathsf{wi}.k_{\mathsf{V}}) \leftarrow \mathsf{WI.Setup}(1^\kappa)$ and $\sigma \overset{\$}{\leftarrow} \{0,1\}^{2n}$. It sets $\mathsf{crs} := (\mathsf{wi.crs}, \sigma)$ and $k_{\mathsf{V}} := \mathsf{wi}.k_{\mathsf{V}}$, and outputs $(\mathsf{crs}, k_{\mathsf{V}})$.

$\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$**:** This algorithm does the following.

1. Defines $\mathcal{L}^\vee := \mathcal{L} \vee \{\exists \mathsf{seed}$ such that $\mathsf{PRG}(\mathsf{seed}) = \sigma\}$. That is, (x, σ) is an instance of language $\mathcal{L}^\vee$. Note that $R^\vee := \{((x,\sigma), w) \mid$ either $(x,w) \in R_{\mathcal{L}}$ or $\mathsf{PRG}(w) = \sigma\}$ is a NP-relation.
2. Generates $\mathsf{wi}.\pi \leftarrow \mathsf{WI.Prove}(\mathsf{wi.crs}, (x,\sigma), w)$. This is possible since Π_{wi} is an NIWI proof system for all NP-languages.
3. Outputs a proof $\pi := \mathsf{wi}.\pi$.

$\mathsf{Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi) \to \top$ **or** $\bot$**:** This algorithm parses $\mathsf{crs} = (\mathsf{wi.crs}, \sigma)$, $k_{\mathsf{V}} = \mathsf{wi}.k_{\mathsf{V}}$, and $\pi = \mathsf{wi}.\pi$, and verifies that $\top = \mathsf{WI.Verify}(\mathsf{wi.crs}, \mathsf{wi}.k_{\mathsf{V}}, (x,\sigma), \mathsf{wi}.\pi)$ for $\mathcal{L}^\vee$. If it does not hold, then outputs $\bot$.

Lemma 3.33. Π_{zk} *satisfies completeness.*

Proof of Lemma 3.33. If $x \in \mathcal{L}$, then the prover has a witness w such that $(x,w) \in \mathcal{R}_{\mathcal{L}}$. Therefore, the completeness follows from the completeness of Π_{wi}. □

Lemma 3.34. Π_{zk} *satisfies soundness.*

Proof of Lemma 3.34. By simple counting argument, $\sigma \overset{\$}{\leftarrow} \{0,1\}^{2n}$ is not in the range of PRG except probability 2^{-n} since the seed length is n. Therefore, except negligible probability, there is no witness for language $\mathcal{L}_g := \{\exists \mathsf{seed}$ such that $\mathsf{PRG}(\mathsf{seed}) = \sigma\}$. That is, except negligible probability, $(x,w) \notin \mathcal{L}^\vee$ since $x \notin \mathcal{L}$. By the soundness of Π_{wi}, the soundness of Π_{zk} follows. □

Lemma 3.35. Π_{zk} *satisfies adaptive multi-theorem (programmable CRS) zero-knowledge.*

Proof of Lemma 3.35. We construct a simulator $\mathsf{zk}.\mathcal{S} = (\mathsf{zk}.\mathcal{S}_1, \mathsf{zk}.\mathcal{S}_2)$ as follows.

1. $\mathsf{zk}.\mathcal{S}_1(1^\kappa)$ generates $(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}})$ as follows.
   - Runs $(\mathsf{wi.crs}, \mathsf{wi}.k_{\mathsf{V}}) \leftarrow \mathsf{WI.Setup}(1^\kappa)$
   - Samples $\mathsf{seed} \overset{\$}{\leftarrow} \{0,1\}^n$, computes $\sigma := \mathsf{PRG}(\mathsf{seed})$

- Sets $\mathsf{crs} := (\mathsf{wi.crs}, \sigma)$, $k_{\mathsf{V}} := \mathsf{wi}.k_{\mathsf{V}}$, and $\tau_{\mathsf{V}} := \mathsf{seed}$.

2. $\mathsf{zk}.\mathcal{S}_1$ sends $(\mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$.
3. When $\mathcal{A}$ sends a query $(x_i, w_i) \in \mathcal{R}_{\mathcal{L}}$ to the oracle, $\mathsf{zk}.\mathcal{S}_2(\mathsf{crs}, k_{\mathsf{V}}, \tau_{\mathsf{V}}, x_i)$ simulates a proof π_i as follows.
   - Runs $\mathsf{wi}.\pi_{\mathsf{seed}} \leftarrow \mathsf{WI.Prove}(\mathsf{wi.crs}, (x_i, \sigma), \mathsf{seed})$. That is, $\mathsf{zk}.\mathcal{S}_2$ uses seed as a witness for $\mathcal{L}^{\vee}$. This is a valid witness since $\mathsf{PRG}(\mathsf{seed}) = \sigma$ by the definition of $\mathsf{zk}.\mathcal{S}_1$ above.
   - Sets $\pi_i := \mathsf{wi}.\pi_{\mathsf{seed}}$.

In the following, we prove that the simulated proofs are indistinguishable from real ones. Suppose that $\mathcal{A}$ distinguishes simulated and real proofs. Then we construct a distinguisher $\mathcal{B}$ that breaks the witness indistinguishability of Π_{wi} as follows.

First, $\mathcal{B}$ is given $(1^{\kappa}, \mathsf{wi.crs}, \mathsf{wi}.k_{\mathsf{V}})$ from the experiment in Definition 3.4. It samples $\mathsf{seed} \xleftarrow{\$} \{0,1\}^n$, sets $\sigma := \mathsf{PRG}(\mathsf{seed})$, $\mathsf{crs} := (\mathsf{wi.crs}, \sigma)$ and $k_{\mathsf{V}} := \mathsf{wi}.k_{\mathsf{V}}$, and runs $\mathcal{A}$ on input $(\mathsf{crs}, k_{\mathsf{V}})$. When $\mathcal{A}$ queries $(x_i, w_i) \in \mathcal{R}_{\mathcal{L}}$ to its oracle, $\mathcal{B}$ queries $((x_i, \sigma), w_i, \mathsf{seed})$ to its own oracle to get $\mathsf{wi}.\pi_i$ and returns $\mathsf{wi}.\pi_i$ to $\mathcal{A}$ as a response by the oracle. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

This completes the description of $\mathcal{B}$. First, we remark that $\mathsf{wi}.\pi_i \neq \bot$ in each query since we have $((x_i, \sigma), w_i) \in \mathcal{L}^{\vee}$ and $((x_i, \sigma), \mathsf{seed}) \in \mathcal{L}^{\vee}$. Then it is easy to see that $\mathcal{B}$ perfectly simulates the experiment where $\mathcal{A}$ gets real proofs if the coin chosen in the witness indistinguishability experiment $\mathcal{B}$ is involved is equal to 0, and $\mathcal{B}$ perfectly simulates the experiment where $\mathcal{A}$ gets simulated proofs otherwise. Therefore if $\mathcal{A}$ distinguishes real and simulated proofs, then $\mathcal{B}$ breaks the witness indistinguishability of Π_{wi}. This completes the proof of Lemma 3.35.

□

By Lemmata 3.33 to 3.35, we obtain Theorem 3.32. Finally, by Theorems 3.8 and 3.32, we obtain Theorem 3.1 presented at the beginning of Section 3.

4 Constructing HomSig from ABE-Simulation Paradigm

In this section, we construct a context-hiding HomSig for $\mathbf{NC}^1$ from a new non-static (q-type) assumption on pairing groups that we call the CDHER assumption. Specifically, we first construct a new ABE scheme from the same assumption and then apply the (semi-generic) conversion sketched in Section 1.2. We directly give a construction of HomSig instead of constructing it via the new ABE. Using the transformation by Kim and Wu [KW18a] (which is described in Appendix C.2), we obtain a DP-NIZK from the same assumption. The resulting DP-NIZK has a compact proof size, i.e., $|C| + \mathsf{poly}(\kappa)$, where $|C|$ denotes the size of the circuit that computes the relation being proved.

4.1 Preparation: Bilinear Maps and Monotone Span Programs

Before providing the concrete construction, we first prepare the hardness assumption and tools that will be used throughout this section.

Let BGGen be a PPT algorithm that on input 1^{κ} returns a description $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, g, e(\cdot,\cdot))$ of symmetric pairing groups where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, g is the generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map.

Definition 4.1 ((n,m)-Computational Diffie-Hellman Exponent and Ratio Assumption). *Let* BGGen *be a group generator and* $n := n(\kappa) = \mathsf{poly}(\kappa)$, $m := m(\kappa) = \mathsf{poly}(\kappa)$. *We say that the* (n,m)-decisional Diffie-Hellman exponent and ratio (CDHER) assumption *holds with respect to* BGGen*, if for all PPT adversaries* $\mathcal{A}$*, we have*

$$\Pr\left[\mathcal{A}(\mathcal{G}, \Psi) \to e(g,g)^{sa^{m+1}}\right] = \mathsf{negl}(\kappa)$$

where $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, g, e(\cdot,\cdot)) \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $s, a, b_1, \ldots, b_n, c_1, \ldots c_n \xleftarrow{\$} \mathbb{Z}_p^*$, *and*

$$\Phi := \left(\begin{array}{cccc} \left\{g^{a^j}\right\}_{j\in[m]}, & \{g^{c_i}\}_{i\in[n]}, & \left\{g^{a^j/b_i}\right\}_{\substack{i\in[n],j\in[2m]\\ j\neq m+1}}, & \left\{g^{a^{m+1}c_{i'}/b_ic_i}\right\}_{i,i'\in[n],i\neq i'}, \\ & \{g^{ac_i}\}_{i\in[n]}, & \left\{g^{a^j/b_ic_i}\right\}_{i\in[n],j\in[2m+1]}, & \left\{g^{a^jc_{i'}/b_i}\right\}_{i,i'\in[n],j\in[m]}, \\ g^s, & \{g^{sb_i}\}_{i\in[n]}, & \left\{g^{sa^{m+1}b_i/b_{i'}c_{i'}}\right\}_{i,i'\in[n],i\neq i'}, & \left\{g^{sa^jb_i/b_{i'}}\right\}_{\substack{i,i'\in[n],j\in[m]\\ i\neq i'}} \end{array} \right).$$

As a sanity check, we show the following lemma, which asserts that the assumption holds on the generic group model introduced by Shoup [Sho97].

Lemma 4.2. *The* (n, m)*-CDHER assumption holds on the generic group model for any* $n, m = \mathsf{poly}(\kappa)$.

Proof. We can prove somewhat stronger claim. Namely, the decision version of the (n, m)-CDHER assumption, where the adversary should distinguish $e(g, g)^{sa^{m+1}}$ from a random group element in $\mathbb{G}_T$, holds on the generic group model. The decision version of the above assumption can be seen as an instance of $\mathbb{G}_T$-monomial assumption introduced by Rouselakis and Waters [RW13]. Thus, by [RW12, Corollary D.4], it suffices to show that pairing result of any two elements from Φ is not (symbolically) equivalent to $e(g, g)^{sa^{m+1}}$. We first observe that the only elements in Φ that contain "s" in the exponent are g^s, g^{sb_i}, $g^{sa^{m+1}b_i/b_{i'}c_{i'}}$, and $g^{sa^jb_i/b_{i'}}$, where $i \neq i'$ for the latter two terms. To obtain $e(g, g)^{sa^{m+1}}$ as a pairing result, one of the input to the pairing operation should be one of these terms. Therefore, the other input should be either $g^{a^{m+1}}$, g^{a^{m+1}/b_i}, $g^{b_{i'}c_{i'}/b_i}$, or $g^{a^{m+1-j}b_{i'}/b_i}$, where $i \neq i'$ for the latter two terms. However, these terms are not given in the problem instance Φ. □

We define a slightly simplified version of (monotone) span programs below.

Definition 4.3 (Monotone Span Program). *A (monotone) span program for universe* $[n]$ *is a matrix* $\mathbf{M}$*, where* $\mathbf{M}$ *is an* $n \times m$ *matrix over* $\mathbb{Z}_p$*. Given* $\mathbf{y} = (y_1, \ldots, y_n) \in \{0, 1\}^n$*, we say that*

$$\mathbf{y} \textit{ satisfies } \mathbf{M} \textit{ iff } \mathbf{1} \in \mathrm{span}\,\langle \mathbf{M}_I \rangle\,.$$

Here $\mathbf{1} = (1, 0, \ldots, 0) \in \mathbb{Z}_p^{1\times m}$ *is a row vector;* $\mathbf{M}_I$ *denotes the matrix obtained by removing the* j*-th row of* $\mathbf{M}$ *for* j *such that* $j \notin I$ *for* $I := \{i \in [n] \mid y_i = 0\}$[8]*; and* span *refers to* $\mathbb{Z}_p$*-linear span of row vectors.*

That is, $\mathbf{y}$ satisfies $\mathbf{M}$ iff there exist coefficients $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that

$$\sum_{i\in I} w_i \mathbf{M}_i = \mathbf{1},$$

where $\mathbf{M}_i$ denotes the i-th row vector of $\mathbf{M}$. Observe that the coefficients $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ can be computed in time polynomial in the size of $\mathbf{M}$ via Gaussian elimination.

Note that we adopt a slightly non-standard definition of the monotone span program, in that we do not allow the program to read the same input bit multiple times. This is for the sake of the brevity and this limitation can be removed by blowing up the matrices as well as inputs by a polynomial factor.

The following lemma is taken from [GPSW06b].

Lemma 4.4 ([GPSW06b, Proposition 1]). *If a vector* $\mathbf{y} \in \{0, 1\}^n$ *does not satisfy a (monotone) span program* $\mathbf{M} \in \mathbb{Z}_p^{n\times m}$*, then there exists an efficiently computable vector* $\mathbf{d} = (d_1, \ldots, d_m) \in \mathbb{Z}_p^m$ *such that* $\mathbf{M}_I\mathbf{d}^\top = \mathbf{0}$ *and* $d_1 = -1$*, where* $I := \{i \in [n] \mid y_i = 0\}$.

It is well known that $\mathbf{NC}^1$ circuits can be represented as a polynomial-sized Boolean formulae. Furthermore, any polynomial-sized Boolean formulae can be converted into an equivalent monotone span program (See e.g., Appendix G of [LW11]). Combining them, we have the following lemma, which allows us to use (monotone) span programs as $\mathbf{NC}^1$ circuits instead.

[8] We note that our definition of I here is somewhat non-standard. Usually, we define I as $I := \{i \in [n] \mid y_i = 1\}$. This change is introduced because it slightly simplifies our presentation and is not essential.

Lemma 4.5. *Let $d = d(\kappa), \ell = \ell(\kappa)$, and $s = s(\kappa)$ be integers. There exist integer parameters $n = n(d, \ell, s)$ and $m = m(d, \ell, s)$ and* deterministic *algorithms* EncInp *and* EncCir *with the following properties.*

- $\mathsf{EncInp}(\mathbf{x}) \to \mathbf{y} \in \{0,1\}^n$*, where* $\mathbf{x} \in \{0,1\}^\ell$.
- $\mathsf{EncCir}(C) \to \mathbf{M} \in \{-1,0,1\}^{n \times m}$*, where* $C : \{0,1\}^\ell \to \{0,1\}$ *is a circuit with depth and size bounded by* d *and* s*, respectively.*

We have that $\mathbf{y}$ *satisfies the span program* $\mathbf{M}$ *over* $\mathbb{Z}_p$ *if and only if* $C(\mathbf{x}) = 0$ *for any prime modulus* $p > 3$*. Furthermore, the running time of* EncInp *and* EncCir *is* $\mathsf{poly}(\ell, s, 2^d)$*. In particular, if* C *is a polynomial-sized circuit with logarithmic depth (i.e., if the circuit is in* $\mathbf{NC}^1$*),* EncInp *and* EncCir *run in polynomial time. In particular, we have* $n = \mathsf{poly}(\kappa)$ *and* $m = \mathsf{poly}(\kappa)$ *in this case.*

4.2 Compact Construction from the CDHER Assumption

In this section, we give a HomSig scheme for $\mathbf{NC}^1$ circuits from CDHRE assumption on bilinear groups. The HomSig scheme achieves very short (evaluated) signatures, which consist of only constant number of group elements. Combining the construction with Theorem 2.21, We can obtain DP-NIZK scheme with short proofs. In particular, when the proven relation is described by a leveled circuit, the proof size can be smaller than the circuit size (See Appendix C.4). Furthermore, the scheme can be extended to the multi-data setting and provide the online-offline efficiency. We refer to Appendix D for the details.

Description of the Function Class. Let $d(\kappa) = O(\log \kappa)$, $\ell = \mathsf{poly}(\kappa)$, and $s = \mathsf{poly}(\kappa)$. The circuit class dealt with by our FHS scheme is denoted as $\mathcal{C}^{\mathbf{NC}^1} = \{\mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}\}_{\kappa \in \mathbb{N}}$, where $\mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ is a set of circuits whose input lengths are ℓ, and depths and sizes are bounded by d and s, respectively. We also define a circuit class $\tilde{\mathcal{C}}^{\mathbf{NC}^1} = \{\tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}\}_{\kappa \in \mathbb{N}}$ associated with $\mathcal{C}^{\mathbf{NC}^1}$ as

$$\tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s} = \{\tilde{C}_z(\cdot) = (C(\cdot) \stackrel{?}{=} z) \mid \forall z \in \{0,1\}, \forall C \in \mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}\}.$$

More specifically, $\tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ is a set of circuits with input length ℓ such that a circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ on input $\mathbf{x} \in \{0,1\}^\ell$ outputs 1 if and only if circuit $C \in \mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ outputs z on input $\mathbf{x}$. Since $(\mathbf{x} \stackrel{?}{=} \mathbf{y})$ can be expressed by a constant-size circuit, every circuit in $\tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ have input length ℓ, and the depths and sizes are bounded by $d + O(1)$ and $s + O(1)$, respectively. Then, by Lemma 4.5, there exist $n(\kappa) = \mathsf{poly}(\kappa)$ and $m(\kappa) = \mathsf{poly}(\kappa)$ such that $\mathsf{EncInp}(x) \in \{0,1\}^n$ for any $x \in \{0,1\}^\ell$ and $\mathsf{EncCir}(C) \in \{-1,0,1\}^{n \times m}$ for any $C \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$. Furthermore, all of these values can be computed in time $\mathsf{poly}(\kappa)$. In the following, since we fix κ, d, ℓ, and s in the construction, we drop the subscript and denote $\mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}, \tilde{\mathcal{C}}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}$ as $\mathcal{C}^{\mathbf{NC}^1}, \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ for notational convenience.

Construction. In the following each signature spaces Σ_{Fresh} and Σ_{Evaled} are set as $\mathbb{Z}_p$ and $\mathbb{G}^3 \times \{0,1\}$, respectively.

$\mathsf{HS.KeyGen}(1^\kappa, 1^\ell)$: On input the security parameter 1^κ and the message length 1^ℓ, sample the group description $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, g, e(\cdot,\cdot)) \leftarrow \mathsf{BGGen}(1^\kappa)$. Then sample $a, u \leftarrow \mathbb{Z}_p$ and $b_i, c_i \leftarrow \mathbb{Z}_p^*$ for $i \in [n]$ and output

$$\mathsf{vk} = \begin{pmatrix} \left\{g^{a^j}\right\}_{j\in[m]}, & \{g^{c_i}\}_{i\in[n]}, & \left\{g^{a^j/b_i}\right\}_{\substack{i\in[n],j\in[2m]\\ j\neq m+1}}, & \left\{g^{a^{m+1}c_{i'}/b_ic_i}\right\}_{i,i'\in[n],i\neq i'} \\ g^u, & \{g^{ac_i}\}_{i\in[n]}, & \left\{g^{a^j/b_ic_i}\right\}_{i\in[n],j\in[2m+1]}, & \left\{g^{a^jc_{i'}/b_i}\right\}_{i,i'\in[n],j\in[m]} \end{pmatrix} \tag{1}$$

and

$$\mathsf{sk} = \big(a, u, \{b_i, c_i\}_{i\in[n]}\big).$$

For notational simplicity, we assume the group description $\mathcal{G}$ is implicitly included both in vk and sk.

$\mathsf{HS.Sign}(\mathsf{sk}, \mathbf{x} = (x_1, \cdots, x_\ell))$: On input $\mathbf{x} \in \{0,1\}^\ell$, run $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$, and compute $\tilde{u} \in \mathbb{Z}_p$ as

$$\tilde{u} = u - \sum_{i \in [n]} y_i \cdot \left(a^{m+1}/b_i c_i\right),$$

where $y_i \in \{0,1\}$ is the i-th bit of $\mathbf{y}$. Finally, output $\boldsymbol{\sigma} = \tilde{u} \in \Sigma_{\mathsf{Fresh}}$.

$\mathsf{HS.Eval}(\mathsf{vk}, C, \mathbf{x}, \boldsymbol{\sigma})$: If $\mathbf{x} \notin \{0,1\}^\ell$, $C \notin \mathcal{C}^{\mathbf{NC}^1}$ or $\boldsymbol{\sigma} = \tilde{u} \notin \Sigma_{\mathsf{Fresh}}$, abort. Otherwise, compute $z = C(\mathbf{x}) \in \{0,1\}$ and construct the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ defined above. Here, we have $\tilde{C}_z(\mathbf{x}) = 1$ by construction. Then, run $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$ and $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M} \in \mathbb{Z}_p^{n \times m}$. By Lemma 4.5, $\mathbf{y}$ *does not* satisfy the span program $\mathbf{M}$ since $\tilde{C}_z(\mathbf{x}) = 1$. Then find a vector $\mathbf{d} = (d_1, \ldots, d_m) \in \mathbb{Z}_p^m$ such that $d_1 = -1$ and $\langle \mathbf{M}_i, \mathbf{d} \rangle = 0$ for all $i \in [n]$ satisfying $y_i = 0$, where $\mathbf{M}_i$ is the i-th row of $\mathbf{M}$. Note that such a vector exists and can be found efficiently due to Lemma 4.4. Then pick $\tilde{r} \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$K_1 = g^{\tilde{r}} \cdot \prod_{j \in [m]} \left(g^{a^{m+1-j}}\right)^{d_j} \cdot \prod_{i \in [n]} (g^{c_i})^{-\langle \mathbf{M}_i, \mathbf{d} \rangle} \quad \text{and} \tag{2}$$

$$K_2 = (g^a)^{\tilde{r}} \cdot \prod_{j \in [2,m]} (g^{a^{m+2-j}})^{d_j} \cdot \prod_{i \in [n]} (g^{a c_i})^{-\langle \mathbf{M}_i, \mathbf{d} \rangle}. \tag{3}$$

Note that the above terms can be efficiently computed as linear combinations of the group elements in the verification key vk. Then compute

$$\begin{aligned}
L_1 &:= K_1^{\tilde{u}} \cdot \left(\prod_{i \in [n]} \left(g^{a^{m+1}/b_i c_i}\right)^{y_i} \cdot \prod_{i \in [n], j \in [m]} \left(g^{a^j/b_i}\right)^{M_{i,j}} \right)^{\tilde{r}}, \\
L_2 &:= \prod_{i \in [n], j \in [m]} \left(g^{a^{2m+2-j}/b_i c_i}\right)^{d_j y_i} \cdot \prod_{i,i' \in [n], j \in [m]} \left(g^{a^j c_{i'}/b_i}\right)^{-\langle \mathbf{M}_{i'}, \mathbf{d} \rangle M_{i,j}}, \\
L_3 &:= \prod_{\substack{i,i' \in [n] \\ i \neq i'}} \left(g^{a^{m+1} c_{i'}/b_i c_i}\right)^{-\langle \mathbf{M}_{i'}, \mathbf{d} \rangle y_i} \cdot \prod_{\substack{i \in [n], j, j' \in [m], \\ j \neq j'}} \left(g^{a^{m+1-j'+j}/b_i}\right)^{M_{i,j} d_{j'}}, \quad \text{and} \\
K_3 &:= L_1 \cdot L_2 \cdot L_3
\end{aligned} \tag{4}$$

Note that all of them can be efficiently computed as linear combinations of the group elements in the verification key vk. Finally, output $\sigma = (K_1, K_2, K_3, z) \in \Sigma_{\mathsf{Evaled}}$.

$\mathsf{HS.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma})$: Output $\perp$ if $\mathbf{x} \notin \{0,1\}^\ell$ or $\boldsymbol{\sigma} = \tilde{u} \notin \Sigma_{\mathsf{Fresh}}$. Otherwise, first run $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$. Then, check the following condition:

$$g^u \stackrel{?}{=} g^{\tilde{u}} \cdot \prod_{i \in [n]} \left(g^{a^{m+1}/b_i c_i}\right)^{y_i},$$

where $y_i \in \{0,1\}$ is the i-th bit of $\mathbf{y}$. If it holds output $\top$, otherwise output $\perp$.

$\mathsf{HS.VerifyEvaled}(\mathsf{vk}, C, z, \sigma)$: Parse $\sigma = (K_1, K_2, K_3, z')$ and output $\perp$ if any of the following holds: $z \notin \{0,1\}$, $z \neq z'$, $\sigma \notin \Sigma_{\mathsf{Evaled}}$, and $C \notin \mathcal{C}^{\mathbf{NC}^1}$. Otherwise, construct the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ as defined above. Then, run $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M} \in \mathbb{Z}_p^{n \times m}$ and check the following conditions:

$$e\left(K_1,\ g^u \cdot \prod_{i \in [n], j \in [m]} \left(g^{a^j/b_i}\right)^{M_{i,j}}\right) \stackrel{?}{=} e(K_3, g), \qquad e(g, K_2) \cdot e(g^a, K_1)^{-1} \stackrel{?}{=} e\left(g^a, g^{a^m}\right),$$

where $M_{i,j}$ is the (i,j)-th entry of the matrix $\mathbf{M}$. If the above equations hold, output $\top$. Otherwise output $\perp$.

Context-Hiding. We first prove the context-hiding of the scheme and then prove the correctness of the scheme, since this makes the proofs slightly simpler.

Theorem 4.6. *Our construction is perfectly context-hiding.*

Proof. To show the context-hiding property, we first construct the homomorphic signature simulator $\mathsf{HS.Sim}$ as follows:

$\mathsf{HS.Sim}(\mathsf{vk}, \mathsf{sk}, C, z)$: On input a circuit $C \in \mathcal{C}^{\mathbf{NC}^1}$ and a message $z \in \{0,1\}$, it first constructs the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ associated to circuit C. Then, it computes $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M} \in \mathbb{Z}_p^{n \times m}$. It then picks $r \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$K_1 := g^r, \quad K_2 := g^{a^{m+1}} \cdot (g^a)^r, \quad K_3 := \left(g^u \cdot \prod_{i \in [n], j \in [m]} \left(g^{a^j / b_i} \right)^{M_{i,j}} \right)^r \tag{5}$$

and outputs $\sigma = (K_1, K_2, K_3, z)$.

We now proceed to show that this simulator $\mathsf{HS.Sim}$ satisfies the required conditions. First, let $\mathbf{x} \in \{0,1\}^\ell$ be an arbitrary input such that $\tilde{C}_z(\mathbf{x}) = 1$, and let $y_i \in \{0,1\}$ be the i-th bit of $\mathbf{y} = \mathsf{EncInp}(\mathbf{x})$. Furthermore, let $\mathbf{d} \in \mathbb{Z}_p^m$ as defined in $\mathsf{HS.Eval}$. Then, let us rewrite r and u in Eq. (5) as

$$r = \tilde{r} + \sum_{j \in [m]} d_j a^{m+1-j} - \sum_{i \in [n]} \langle \mathbf{M}_i, \mathbf{d} \rangle c_i \qquad \text{and} \qquad u = \tilde{u} + \sum_{i \in [n]} y_i \cdot \left(a^{m+1} / b_i c_i \right),$$

where $\mathbf{M}_i$ is the i-th row of $\mathbf{M}$. In the following, we show that K_1, K_2, and K_3 are distributed exactly as in Eq. (2), (3), and (4) where the probability is taken over the randomness of r. Here, recall that all other elements including u, hence $\tilde{u}$, are fixed by the definition of context-hiding. First of all, it is easy to check that the above K_1 is distributed identically to Eq. (2), since r is distributed uniformly at random if and only if so is $\tilde{r}$. Now, since K_2 and K_3 are uniquely defined once K_1 is fixed, it remains to check that the values of K_2 and K_3 conform to Eq. (3) and (4) for fixed r. This can be checked easily for K_2 since we have

$$\begin{aligned} \log_g K_2 &= a^{m+1} + ar \\ &= a^{m+1} + a \left(\tilde{r} + \sum_{j \in [m]} d_j a^{m+1-j} - \sum_{i \in [n]} \langle \mathbf{M}_i, \mathbf{d} \rangle c_i \right) \\ &= \tilde{r} a + \sum_{j \in [2,m]} d_j a^{m+2-j} - \sum_{i \in [n]} \langle \mathbf{M}_i, \mathbf{d} \rangle a c_i, \end{aligned}$$

where the last equation follows from $d_1 = -1$. Note that the term a^{m+1} cancels out here.

We need some more work for K_3. We have

$$\begin{aligned} & \log_g K_3 \\ = \; & r \left(u + \sum_{i \in [n], j \in [m]} M_{i,j} a^j / b_i \right) \\ = \; & \left(\tilde{r} + \underbrace{\sum_{j \in [m]} d_j a^{m+1-j} - \sum_{i \in [n]} \langle \mathbf{M}_i, \mathbf{d} \rangle c_i}_{:= \Phi_1} \right) \cdot \left(\tilde{u} + \underbrace{\sum_{i \in [n]} y_i a^{m+1} / b_i c_i + \sum_{i \in [n], j \in [m]} M_{i,j} a^j / b_i}_{:= \Phi_2} \right) \\ = \; & \tilde{u}(\tilde{r} + \Phi_1) + \tilde{r}\Phi_2 + \Phi_1 \Phi_2 \\ = \; & \underbrace{\tilde{u} r + \tilde{r} \Phi_2}_{:= \Phi_3} + \Phi_1 \Phi_2 \end{aligned}$$

We can observe that $g^{\Phi_3} = L_1$. We next expand $\Phi_1\Phi_2$ and show that $g^{\Phi_1\Phi_2} = L_2L_3$, which concludes the proof. Before doing so, we define

$$\Phi_{1,1} := \sum_{j\in[m]} d_j a^{m+1-j}, \qquad \Phi_{1,2} := -\sum_{i\in[n]} \langle \mathbf{M}_i, \mathbf{d}\rangle c_i,$$
$$\Phi_{2,1} := \sum_{i\in[n]} y_i a^{m+1}/b_i c_i, \qquad \Phi_{2,2} := \sum_{i\in[n],j\in[m]} M_{i,j} a^j/b_i.$$

It is readily seen that $\Phi_i = \Phi_{i,1} + \Phi_{i,2}$ for $i = 1, 2$ and thus $\Phi_1\Phi_2 = \Phi_{1,1}\Phi_{2,1} + \Phi_{1,1}\Phi_{2,2} + \Phi_{1,2}\Phi_{2,1} + \Phi_{1,2}\Phi_{2,2}$. We have

$$\begin{aligned}
\Phi_{1,1}\Phi_{2,1} + \Phi_{1,2}\Phi_{2,2} &= \sum_{i\in[n],j\in[m]} d_j y_i \left(a^{2m+2-j}/b_i c_i\right) - \sum_{i,i'\in[n],j\in[m]} \langle \mathbf{M}_{i'}, \mathbf{d}\rangle M_{i,j} \left(a^j c_{i'}/b_i\right), \\
\Phi_{1,2}\Phi_{2,1} &= \left(-\sum_{i'\in[n]} \langle \mathbf{M}_{i'}, \mathbf{d}\rangle c_{i'}\right)\left(\sum_{i\in[n]} y_i a^{m+1}/b_i c_i\right) \\
&= -\sum_{\substack{i,i'\in[n]\\ i\neq i'}} \langle \mathbf{M}_{i'}, \mathbf{d}\rangle y_i \left(a^{m+1} c_{i'}/b_i c_i\right) - \sum_{i\in[n]} \langle \mathbf{M}_i, \mathbf{d}\rangle y_i \left(a^{m+1}/b_i\right)
\end{aligned}$$

and

$$\begin{aligned}
\Phi_{1,1}\Phi_{2,2} &= \left(\sum_{j'\in[m]} d_{j'} a^{m+1-j'}\right) \cdot \left(\sum_{i\in[n],j\in[m]} M_{i,j} a^j/b_i\right) \\
&= \sum_{\substack{i\in[n],j,j'\in[m],\\ j\neq j'}} M_{i,j} d_{j'} \left(a^{m+1-j'+j}/b_i\right) + \sum_{i\in[n],j\in[m]} M_{i,j} d_j \left(a^{m+1}/b_i\right) \\
&= \sum_{\substack{i\in[n],j,j'\in[m],\\ j\neq j'}} M_{i,j} d_{j'} \left(a^{m+1-j'+j}/b_i\right) + \sum_{i\in[n]} \langle \mathbf{M}_i, \mathbf{d}\rangle \left(a^{m+1}/b_i\right) \\
&= \sum_{\substack{i\in[n],j,j'\in[m],\\ j\neq j'}} M_{i,j} d_{j'} \left(a^{m+1-j'+j}/b_i\right) + \sum_{i\in[n]} \langle \mathbf{M}_i, \mathbf{d}\rangle y_i \left(a^{m+1}/b_i\right),
\end{aligned}$$

where in the last equation we used $y_i \in \{0, 1\}$ and $\langle \mathbf{M}_i, \mathbf{d}\rangle = 0$ if $y_i = 0$. These imply that

$$\Phi_{1,2}\Phi_{2,1} + \Phi_{1,1}\Phi_{2,2} = -\sum_{\substack{i,i'\in[n]\\ i\neq i'}} \langle \mathbf{M}_{i'}, \mathbf{d}\rangle y_i \left(a^{m+1} c_{i'}/b_i c_i\right) + \sum_{\substack{i\in[n],j,j'\in[m],\\ j\neq j'}} M_{i,j} d_{j'} \left(a^{m+1-j'+j}/b_i\right),$$

where the terms of the form a^{m+1}/b_i all cancel out here. We can see that $g^{\Phi_{1,1}\Phi_{2,1}+\Phi_{1,2}\Phi_{2,2}} = L_2$ and $g^{\Phi_{1,2}\Phi_{2,1}+\Phi_{1,1}\Phi_{2,2}} = L_3$, which imply $g^{\Phi_1\Phi_2} = L_2L_3$. This concludes the proof of the theorem. □

Correctness. Here, we show that the scheme satisfies signing correctness and evaluation correctness defined in Definition 2.15. The signing correctness is easy to check. The evaluation correctness is also easy to check given the fact that K_1, K_2, and K_3 generated by $\mathsf{HS.Eval}$ are distributed as Eq. (5), which is shown in the proof of Theorem 4.6.

Unforgeability. We finally prove unforgeability of our HomSig scheme. Formally, we prove the following theorem.

Theorem 4.7. *Our construction satisfies (single-shot) selective-unforgeability assuming the hardness of the* (n, m)*-CDHER problem.*

Proof. To prove the theorem, it suffices to show that for any PPT adversary $\mathcal{A}$ against the selective-unforgeability of our homomorphic signature scheme Π_{HS} with advantage ϵ, we can construct a PPT algorithm $\mathcal{B}$ that solves the (n,m)-CDHER problem with the same probability. Then, assuming the hardness of the (n,m)-CDHER problem, we conclude our proof. We give the description of $\mathcal{B}$ in the following.

Setup: The reduction algorithm $\mathcal{B}$ is given the problem instance Ψ of the (n,m)-CDHER problem. Then, $\mathcal{B}$ runs $\mathcal{A}$ on input 1^κ and is given 1^ℓ and the messages $\mathbf{x} \in \{0,1\}^\ell$ from $\mathcal{A}$. The reduction algorithm $\mathcal{B}$ runs $\mathbf{y} \leftarrow \mathsf{EncInp}(\mathbf{x})$, samples $\tilde{u} \leftarrow \mathbb{Z}_p$ and computes g^u as

$$g^u := g^{\tilde{u}} \cdot \prod_{i\in[n]} \left(g^{a^{m+1}/b_i c_i}\right)^{y_i}.$$

It then sets the verification key vk as Eq. (1) using the terms in the problem instance where g^u is set as above. Finally, it gives vk to $\mathcal{A}$. It is easy to see that the distribution of vk is the same as that of the real game. Note that the signing key sk is implicitly set as $(a, u = \tilde{u} + \sum_{i\in[n]} y_i \cdot a^{m+1}/b_i c_i, \{b_i, c_i\}_{i\in[n]})$.

Signing Query: The reduction algorithm $\mathcal{B}$ simply returns $\tilde{u}$ as the signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$. Since the signature is unique once $(\mathsf{vk}, \mathsf{sk}, \mathbf{x})$ are fixed, the signature is distributed as in the real game.

Forgery: At some point, $\mathcal{A}$ outputs $(C^\star, z^\star, \sigma^\star)$ as the forgery. $\mathcal{B}$ aborts and outputs $\perp$ if $\mathcal{A}$ did not win the game. Otherwise, $\mathcal{B}$ parses the signature as $\sigma^\star = (K_1^\star, K_2^\star, K_3^\star, z^\star)$, constructs the circuit $\tilde{C}^\star_{z^\star} \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$, and computes $\mathbf{M}^\star \leftarrow \mathsf{EncCir}(\tilde{C}^\star_{z^\star})$. Since the signature passes the verification, there exists $r^\star \in \mathbb{Z}_p$ and $(K_1^\star, K_2^\star, K_3^\star)$ that satisfies

$$K_1^\star := g^{r^\star}, \quad K_2^\star := g^{a^{m+1}} \cdot (g^a)^{r^\star}, \quad K_3^\star := \left(g^u \cdot \prod_{i\in[n],j\in[m]} \left(g^{a^j/b_i}\right)^{M^\star_{i,j}}\right)^{r^\star}, \tag{6}$$

where $M^\star_{i,j}$ is the (i,j)-th entry of $\mathbf{M}^\star$.

We then extract the answer for the CDHER problem from the forgery. Let us define $I := \{i \in [n] : y_i = 0\}$. We first observe that we can compute $(g^u)^{sb_i}$ for $i \in I$ because we have

$$(g^u)^{sb_i} = \left(g^{\tilde{u}} \cdot \prod_{i'\in[n]} \left(g^{a^{m+1}/b_{i'}c_{i'}}\right)^{y_{i'}}\right)^{sb_i} = \left(g^{sb_i}\right)^{\tilde{u}} \cdot \prod_{i'\in[n]\setminus\{i\}} \left(g^{sa^{m+1}b_i/b_{i'}c_{i'}}\right)^{y_{i'}},$$

where the second equality above follows from $y_i = 0$. We then define

$$s_i^\star := \sum_{j\in[m]} M^\star_{i,j} s a^{j-1}$$

for $j \in [m]$. We then define G_i for $i \in [n]$ as follows:

$$\begin{aligned} G_i &:= (g^a)^{-s_i^\star} \cdot \left(g^u \cdot \prod_{i'\in[n],j\in[m]} \left(g^{a^j/b_{i'}}\right)^{M^\star_{i',j}}\right)^{sb_i} \\ &= \prod_{j\in[m]} \left(g^{sa^j}\right)^{-M^\star_{i,j}} \cdot (g^u)^{sb_i} \cdot \prod_{i'\in[n],j\in[m]} \left(g^{sa^j b_i/b_{i'}}\right)^{M^\star_{i',j}} \\ &= (g^u)^{sb_i} \cdot \prod_{i'\in[n]\setminus\{i\},j\in[m]} \left(g^{sa^j b_i/b_{i'}}\right)^{M^\star_{i',j}}, \end{aligned}$$

where the terms $\{g^{sa^j}\}$ all cancel out in the third equality. We can observe that G_i for $i \in I$ can be efficiently computed as a linear combination of the terms in the problem instance of the CDHER. We therefore can compute

$$e(K_1^\star, G_i) \cdot e(K_3^\star, g^{sb_i})^{-1} = e(g,g)^{-r^\star a s_i^\star}$$

for $i \in I$. Next, since $\mathcal{A}$ outputs a valid forgery, we must have $C^\star(\mathbf{x}) \neq z^\star$. Specifically, we have $\tilde{C}^\star_{z^\star}(\mathbf{x}) = 0$. By Lemma 4.5, this implies that $\mathbf{y}$ satisfies the span program $\mathbf{M}^\star$. Hence, by Definition 4.3, there exists an efficiently computable coefficients $\{w_i^\star\}_{i\in I}$ such that $\sum_{i\in I} w_i^\star \mathbf{M}_i^\star = \mathbf{1}$. We observe that

$$\sum_{i\in I} w_i^\star s_i^\star = \sum_{i\in I} w_i^\star \left(\sum_{j\in[m]} M_{i,j}^\star s a^{j-1}\right) = \left(\sum_{i\in I} w_i^\star \mathbf{M}_i^\star\right)\cdot (s, sa, \ldots, sa^{m-1})^\top = s$$

holds for such $\{w_i^\star\}_{i\in I}$. $\mathcal{B}$ then computes

$$e(K_2^\star, g^s)\cdot \prod_{i\in I}\left(e(g,g)^{-r^\star a s_i^\star}\right)^{w_i^\star} = e(g,g)^{sa^{m+1}}\cdot e(g,g)^{sar^\star}\cdot e(g,g)^{-r^\star a\sum_{i\in I} w_i^\star s_i^\star} = e(g,g)^{sa^{m+1}},$$

where g^s is taken from the problem instance. Finally, $\mathcal{B}$ outputs $e(g,g)^{sa^{m+1}}$ as the answer to the (n,m)-CDHER problem. □

Remark 4.8. We can extend the scheme to the multi-data setting that satisfies online-offline verification efficiency. This is based on the observation that it is possible to prove a slightly stronger security notion than selective unforgeability where the adversary can choose $\mathbf{x}$ after seeing vk except for g^u. In more details, we consider a security notion where the setup phase of the selective unforgeability game is changed as follows:

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given 1^κ as input and sends 1^ℓ to the challenger. Then the challenger generates a signing-verification key pair $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$ and gives $\mathsf{vk}\backslash\{g^u\}$ to $\mathcal{A}$. Here, $\mathsf{vk}\backslash\{g^u\}$ denotes all group elements in vk except for g^u. Then, $\mathcal{A}$ submits $\mathbf{x}$ to the challenger. Then, the challenger gives g^u to $\mathcal{A}$.

We call this stronger security notion *partially adaptive unforgeability*. We can see that the above security proof works in the partially adaptive unforgeability setting without any change because only g^u in vk is dependent on the value of $\mathbf{x}$ in the simulation. This observation will be useful when we extend our scheme to the multi-data setting with online-offline efficiency. See Appendix D for the details.

4.3 Compact DP-NIZK

Using the transformation by Kim and Wu [KW18a] (which is described in Appendix C.2) with our HomSig from the CDHER assumption, we obtain a DP-NIZK with a compact proof size from the same assumption.

Theorem 4.9. *If the CDHER assumption holds on a pairing group, then there exists DP-NIZK for all* **NP** *languages with proof size* $|C| + \mathsf{poly}(\kappa)$*, where* $|C|$ *denotes the size of the circuit that computes the relation being proved.*

As far as we know, this is the first DP-NIZK scheme with short proofs without assuming the LWE assumption, fully-homomorphic encryption, indistinguishability obfuscation, or non-falsifiable assumptions. Furthermore, if the proven **NP** relation can be expressed as a leveled circuit, we can reduce the proof size to $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$, where $|w|$ is the length of the witness of the proven relation and a leveled circuit refers to a circuit whose gates can be divided into layers and only gates from the consecutive layers are connected by wires. See Appendix C.4 for the details.

5 HomMAC from Inner Product Functional Encryption

In this section, we give a construction of HomMAC based on a variant of functional encryption for inner-products (IPFE) which we call a *functional encryption for inner-product on exponent* (expIPFE). Namely, we show that an expIPFE scheme that satisfies a property called extractability suffices for constructing statistically unforgeable and computationally context-hiding HomMAC. We also show that such an expIPFE scheme exists under the DDH assumption. As a result, we obtain a statistically unforgeable and computationally context-hiding HomMAC based on the DDH assumption, which yields statistically sound and computationally (non-programmable CRS) zero-knowledge PP-NIZK based on the DDH assumption (over pairing-free groups).

5.1 Functional Encryption for Inner-Product on Exponent

Here, we define functional encryption for inner-product on exponent (expIPFE). In expIPFE, a ciphertext and a secret key are associated with a vector $\mathbf{x} \in \mathbb{Z}_p^\ell$ and $\mathbf{y} \in \mathbb{Z}_p^\ell$ respectively, and if we decrypt the ciphertext with the secret key, then the decryption results in $g^{\langle \mathbf{x}, \mathbf{y} \rangle}$. The formal definition of expIPFEscheme Π_{IPFE} is given as follows.

$\mathsf{Setup}(1^\kappa, 1^\ell) \to (\mathsf{msk}, \mathsf{pp})$**:** The setup algorithm takes as inputs the security parameter 1^κ and the dimension 1^ℓ, and outputs a public parameter pp and a master secret key msk. The public parameter pp specifies an underlying group $\mathbb{G}$ of order p and its generator g.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \to \mathsf{sk}$**:** The key generation algorithm takes as inputs a master secret key msk and a vector $\mathbf{y} = (y_1, ..., y_\ell) \in \mathbb{Z}_p^\ell$, and outputs a secret key sk. Without loss of generality, we assume that sk always contains $\mathbf{y}$.

$\mathsf{Enc}(\mathsf{pp}, \mathbf{x}) \to \mathsf{ct}$**:** The encryption algorithm takes as inputs a public parameter pp and a vector $\mathbf{x} = (x_1, ..., x_\ell) \in \mathbb{Z}_p^\ell$, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) \to \mathsf{d}$**:** The decryption algorithm takes as inputs a public parameter pp, a secret key sk, and a ciphertext ct, and outputs a group element $\mathsf{d} \in \mathbb{G}$.

Correctness. For all $\kappa, n \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \in \mathsf{Setup}(1^\kappa, 1^\ell)$, vectors $\mathbf{x} \in \mathbb{Z}_p^\ell$ and $\mathbf{y} \in \mathbb{Z}_p^\ell$, secret keys $\mathsf{sk} \in \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$, and ciphertexts $\mathsf{ct} \in \mathsf{Enc}(\mathsf{pp}, \mathbf{x})$, we have

$$\mathsf{Dec}(\mathsf{pp}, \mathsf{ct}, \mathsf{sk}) = g^{\langle \mathbf{x}, \mathbf{y} \rangle}.$$

Security. For an adversary $\mathcal{A}$, we consider the following experiment between a challenger and an adversary $\mathcal{A}$.

1. $\mathcal{A}$ is given 1^κ and outputs 1^ℓ.
2. The challenger generates $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$. Let $\mathbb{G}$ and g be a group of order p and its generator specified by pp, respectively.
3. $\mathcal{A}$ is given pp. It is allowed to make arbitrary number of key generation queries. When it makes a key query $\mathbf{y}$, the challenger generates $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$ and returns sk to $\mathcal{A}$. At some point, $\mathcal{A}$ output vectors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$.
4. The challenger randomly picks $\mathsf{coin} \xleftarrow{\$} \{0, 1\}$, generates $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pp}, \mathbf{x}^{(\mathsf{coin})})$.
5. $\mathcal{A}$ is given ct, and allowed to make arbitrary number of key generation queries again. Finally, $\mathcal{A}$ outputs coin'. We say that $\mathcal{A}$ wins if $\mathsf{coin}' = \mathsf{coin}$.

We say that $\mathcal{A}$ is adaptively admissible if for all key queries $\mathbf{y}$ made by $\mathcal{A}_1$ or $\mathcal{A}_2$ and vectors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ output by $\mathcal{A}_1$, we have $\langle \mathbf{x}^{(0)}, \mathbf{y} \rangle = \langle \mathbf{x}^{(1)}, \mathbf{y} \rangle$. We say that Π_{IPFE} is adaptively secure if for all adaptively admissible adversaries $\mathcal{A}$, the probability $|\Pr[\mathcal{A} \text{ wins}] - 1/2|$ is negligible. We say that Π_{IPFE} is adaptively *one-key* secure if for all adaptively admissible adversaries $\mathcal{A}$ that makes at most one key query, the probability $|\Pr[\mathcal{A} \text{ wins}] - 1/2|$ is negligible.

Extractability. We define an additional property called the extractability. Roughly speaking, extractability requires that any (possibly maliciously generated) ciphertext corresponds to a unique message that is consistent to the decryption result by an honestly generated secret key. More formally, the definition is stated as follows. We say that Π_{IPFE} is extractable if there exists possibly unbounded time extractor Ext such that for all κ, $\ell \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \in \mathsf{Setup}(1^\kappa)$, vectors $\mathbf{y} \in \mathbb{Z}_p^\ell$, secret keys $\mathsf{sk} \in \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$, and ciphertexts ct, if $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) \neq \perp$, then $\mathsf{Ext}(\mathsf{msk}, \mathsf{ct})$ outputs some vector $\mathbf{x} \in \mathbb{Z}_p^\ell$ such that $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) = g^{\langle \mathbf{x}, \mathbf{y} \rangle}$ with probability 1.

Remark 5.1. Extractability can be seen as a weaker form of verifiability for FE proposed by Badrinarayanan et al. [BGJS16], which requires that we can publicly verify that a ciphertext is honestly generated. However, their construction of verifiable FE requires non-interactive witness indistinguishable proofs (NIWI), and not meaningful for our purpose to construct PP-NIZK.

The following theorem addresses the existence of expIPFE with extractability.

Theorem 5.2. *There exists an adaptively secure extractable expIPFE under the DDH assumption.*

The proof will appear in Appendix B.1, where we show that the IPFE proposed by Agrawal et al. [ALS16] satisfies the required properties.

5.2 HomMAC from expIPFE

We give a construction of context-hiding HomMAC based on an expIPFE scheme. Before describing the scheme, we prepare a simple lemma.

Lemma 5.3. *There exists a deterministic polynomial-time algorithm* Coefficient *that satisfies the following: for any $p \in \mathbb{N}$, arithmetic circuit f over $\mathbb{Z}_p$ of degree D, $\mathbf{x} = (x_1, ..., x_\ell) \in \mathbb{Z}_p^\ell$ and $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_\ell) \in \mathbb{Z}_p^\ell$,* $\mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$ *outputs $(c_1, ..., c_D)$ such that*

$$f(\sigma_1 Z + x_1, ..., \sigma_\ell Z + x_\ell) = f(x_1, ..., x_\ell) + \sum_{j=1}^{D} c_j Z^j \mod p. \tag{7}$$

where Z is an indeterminate.

Proof. First, we can see that the constant term of $f(\sigma_1 Z + x_1, ..., \sigma_\ell Z + x_\ell)$ is $f(x_1, ..., x_\ell)$ by substituting $Z = 0$. The other coefficients $(c_1, ..., c_D)$ can be computed by simply expanding $f(\sigma_1 Z + x_1, ..., \sigma_\ell Z + x_\ell)$. This can be done in a polynomial time in D. □

Let $\Pi_{\mathsf{IPFE}} = (\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ be an expIPFE scheme. We construct a HomMAC $\Pi_{\mathsf{HA}} = (\mathsf{HA.KeyGen}, \mathsf{HA.Sign}, \mathsf{HA.Eval}, \mathsf{HA.VerifyFresh}, \mathsf{HA.VerifyEvaled})$ for arithmetic circuits of degree at most $D = \mathsf{poly}(\kappa)$ over a message space $\mathcal{X} = \mathbb{Z}_p$ as follows.[9]

$\mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$**:** This algorithm runs $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{msk}_{\mathsf{IPFE}}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^\kappa, 1^D)$. Let $(\mathbb{G}, g, p)$ be a group, its generator, and its order specified by $\mathsf{pp}_{\mathsf{IPFE}}$. It picks $(r_1, ..., r_\ell) \in \mathbb{Z}_p^\ell$ and $s \xleftarrow{\$} \mathbb{Z}_p^*$ and generates $\mathsf{sk}_{\mathsf{IPFE}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}_{\mathsf{IPFE}}, (s, ..., s^D))$. It outputs a verification key $\mathsf{vk} := (\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell))$, an evaluation key $\mathsf{ek} := \mathsf{pp}_{\mathsf{IPFE}}$ and a signing key $\mathsf{sk} = \mathsf{vk}$.

$\mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x} = (x_1, ..., x_\ell))$**:** This algorithm parses $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell)) \leftarrow \mathsf{sk}$, computes $\sigma_i := (r_i - x_i)s^{-1} \mod p$ for $i \in [\ell]$ and outputs $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_\ell)$.

$\mathsf{HA.Eval}(\mathsf{ek}, f, \mathbf{x}, \boldsymbol{\sigma})$**:** This algorithm parses $\mathsf{pp}_{\mathsf{IPFE}} \leftarrow \mathsf{ek}$, $(x_1, ..., x_\ell) \leftarrow \mathbf{x}$ and $(\sigma_1, ..., \sigma_\ell) \leftarrow \boldsymbol{\sigma}$. It computes $(c_1, ..., c_D) \leftarrow \mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$ and $\mathsf{ct} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{pp}_{\mathsf{IPFE}}, (c_1, ..., c_D))$, and outputs $\sigma := \mathsf{ct}$.

$\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma})$**:** This algorithm parses $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell)) \leftarrow \mathsf{vk}$, $(x_1, ..., x_\ell) \leftarrow \mathbf{x}$, and $(\sigma_1, ..., \sigma_\ell) \leftarrow \boldsymbol{\sigma}$. It outputs $\top$ if $\sigma_i = (r_i - x_i)s^{-1} \mod p$ for all $i \in [\ell]$ and $\bot$ otherwise.

$\mathsf{HA.VerifyEvaled}(\mathsf{vk}, f, z, \sigma)$**:** This algorithm parses $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell)) \leftarrow \mathsf{vk}$, and $\mathsf{ct} \leftarrow \sigma$. It computes $T \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{ct}, \mathsf{sk}_{\mathsf{IPFE}})$. Then it outputs $\top$ if $g^{f(r_1,...,r_\ell)-z} = T$ and $\bot$ otherwise.

Correctness. For any $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_\ell)$ generated by $\mathsf{HA.Sign}(\mathsf{sk}, \mathbf{x} = (x_1, ..., x_\ell))$ we have $r_i = \sigma_i s + x_i$ for $i \in [\ell]$. Then by Lemma 5.3 we have $f(r_1, ..., r_\ell) = f(x_1, .., x_\ell) + \sum_{j=1}^{D} c_j s^j \mod p$ for $(c_1, ..., c_D) \leftarrow \mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$. Since σ generated by $\mathsf{HA.Eval}(\mathsf{ek}, f, \mathbf{x}, \boldsymbol{\sigma})$ is an encryption of $(c_1, ..., c_D)$ of Π_{IPFE}, by the correctness of Π_{IPFE}, we have $T = \mathsf{IPFE.Dec}(\sigma, \mathsf{sk}_{\mathsf{IPFE}}) = g^{\langle (c_1,...,c_D),(s,s^2,...,s^D) \rangle} = g^{\sum_{j=1}^{D} c_j s^j}$ where $\mathsf{sk}_{\mathsf{IPFE}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}_{\mathsf{IPFE}}, (s, s^2, ..., s^D))$ as in the key generation algorithm. Since $f(r_1, ..., r_\ell) = f(x_1, .., x_\ell) + \sum_{j=1}^{D} c_j s^j \mod p$, we have $g^{f(r_1,...,r_\ell)-f(x_1,..,x_\ell)} = T$. Thus $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, f, f(x_1, ..., x_\ell), \sigma)$ returns $\top$.

Security. The following theorems address the security of above scheme.

Theorem 5.4. *Our construction satisfies statistical (single-shot) adaptive-unforgeability assuming that Π_{IPFE} is extractable.*

Theorem 5.5. *Our construction is computationally context-hiding assuming that Π_{IPFE} is adaptively one-key secure.*

[9] Strictly speaking, this does not match the definition of HomAuth schemes given in Section 2.3 since the circuit class and message space depend on p, which is specified by the public parameter. We note that this difference does not affect for the application to PP-NIZK.

Proof of Theorem 5.4

Proof. The proof is similar to the security proof of the HomMAC based on PRFs proposed by Catalano and Fiore [CF18] except that we use an extractor of Π_{IPFE}. Suppose that there exists an adversary $\mathcal{A}$ that breaks the unforgeability of Π_{HA}. First, we remark that we can assume that $\mathcal{A}$ does not make a verification query for fresh signatures without loss of generality. This is because if $\mathcal{A}$ can generate a fresh signature that is not given by the challenger and accepted by $\mathsf{HA.VerifyFresh}$, it can also forge an evaluated signature just by evaluating the fresh signature by a projection function. Therefore we can simulate $\mathsf{HA.VerifyFresh}$ by simply returning $\perp$ for all queries except for the signature given by the challenger. Thus we only consider an adversary $\mathcal{A}$ that only makes a verification query for evaluated signatures. Then we consider the following sequence of games between $\mathcal{A}$ and a challenger.

Game_0**:** This game is the original experiment that defines the unforgeability. Specifically, the game is described as follows.

1. $\mathcal{A}$ is given 1^κ and outputs 1^ℓ.
2. The challenger generates $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{msk}_{\mathsf{IPFE}}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^\kappa, 1^D)$, picks $(r_1, ..., r_\ell) \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $s \xleftarrow{\$} \mathbb{Z}_p^*$, generates $\mathsf{sk}_{\mathsf{IPFE}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}_{\mathsf{IPFE}}, (s, ..., s^D))$, and sets $\mathsf{vk} := (\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell))$, $\mathsf{ek} := \mathsf{pp}_{\mathsf{IPFE}}$ and $\mathsf{sk} := \mathsf{vk}$.
3. $\mathcal{A}$ is given ek and outputs $\mathbf{x} = (x_1, ..., x_\ell)$.
4. The challenger computes $\sigma_i := (r_i - x_i)s^{-1} \mod p$ for $i \in [\ell]$.
5. $\mathcal{A}$ is given $\boldsymbol{\sigma} = (\sigma_1, .., \sigma_\ell)$, and allowed to make arbitrary number of verification queries for evaluated signatures. When $\mathcal{A}$ queries (f, z, ct), the challenger returns $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, f, z, \mathsf{ct})$. That is, it computes $T := \mathsf{IPFE.Dec}(\mathsf{ct}, \mathsf{sk}_{\mathsf{IPFE}})$, and returns $\top$ iff $g^{f(r_1,...,r_\ell)-z} = T$ holds.
6. Finally, $\mathcal{A}$ outputs $(f^*, z^*, \mathsf{ct}^*)$. We say that $\mathcal{A}$ wins if $z^* \neq f^*(x_1, ..., x_\ell)$ and $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, f^*, z^*, \mathsf{ct}^*) = \top$.

Game_1**:** This game is the same as the previous game except that the verification oracle $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, \cdot, \cdot, \cdot)$ is modified to the following alternative oracle denoted by $\mathsf{HA.VerifyEvaled}_1$. The oracle $\mathsf{HA.VerifyEvaled}_1$, given a query (f, z, ct), first computes $(c_1, ..., c_D) := \mathsf{Ext}(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{ct})$ and returns $\top$ iff $f(r_1, ..., r_\ell) = z + \sum_{j=1}^{D} c_i s^i \mod p$ where Ext is the extractor for Π_{IPFE}. We note that the winning condition of $\mathcal{A}$ is also modified to use $\mathsf{HA.VerifyEvaled}_1$ instead of $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, \cdot, \cdot, \cdot)$.

Game_2**:** This game is the same as the previous game except that the oracle $\mathsf{HA.VerifyEvaled}_1$ is replaced with $\mathsf{HA.VerifyEvaled}_2$ described in the following. The oracle $\mathsf{HA.VerifyEvaled}_2$, given a query (f, z, ct), first computes $(c_1, ..., c_D) := \mathsf{Ext}(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{ct})$. It also computes $(\hat{c}_1, ..., \hat{c}_D) \leftarrow \mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$. For notational convenience, we let $c_0 := z$ and $\hat{c}_0 := f(x_1, ..., x_\ell)$. It returns $\top$ iff either of the following holds:

1. $(c_0, ..., c_D) = (\hat{c}_0, ..., \hat{c}_D)$.
2. $(c_0, ..., c_D) \neq (\hat{c}_0, ..., \hat{c}_D)$ and $Z := \sum_{j=0}^{D} (c_j - \hat{c}_j)s^j \mod p = 0$.

We note that the winning condition of $\mathcal{A}$ is also modified to use $\mathsf{HA.VerifyEvaled}_2$ instead of $\mathsf{HA.VerifyEvaled}_1$. Furthermore, observe that $\mathsf{HA.VerifyEvaled}_2$ can be simulated without using $(r_1, ..., r_\ell)$.

Game_3**:** This game is the same as the previous game except that $\sigma_1, ..., \sigma_\ell$ are uniformly chosen from $\mathbb{Z}_p$ instead of setting as $\sigma_i := (r_i - x_i)s^{-1} \mod p$ for $i \in [\ell]$. We note that this game still uses $\mathsf{HA.VerifyEvaled}_2$ as a verification oracle. We also note that $(r_1, ..., r_\ell)$ is not used at all in this game.

Game_4**:** This game is the same as the previous game except that the oracle $\mathsf{HA.VerifyEvaled}_2$ is replaced with an alternative oracle $\mathsf{HA.VerifyEvaled}_4$ described below. The oracle $\mathsf{HA.VerifyEvaled}_4$ works similarly to $\mathsf{HA.VerifyEvaled}_2$ described in Game_2 except that we only use the first condition to reply $\top$ described in Game_2. More precisely, The oracle $\mathsf{HA.VerifyEvaled}_4$, given a query (f, z, ct), first computes $(c_1, ..., c_D) := \mathsf{Ext}(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{ct})$. It also computes $(\hat{c}_1, ..., \hat{c}_D) \leftarrow \mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$. For notational convenience, we let $c_0 := z$ and $\hat{c}_0 := f(x_1, ..., x_\ell)$. It returns $\top$ iff $(c_0, ..., c_D) = (\hat{c}_0, ..., \hat{c}_D)$.

This completes the description of games. We denote the event that $\mathcal{A}$ wins in Game_k by T_k for $k = 0, ..., 4$. We prove the following lemmas.

Lemma 5.6. $\Pr[\mathsf{T}_0] = \Pr[\mathsf{T}_1]$.

Proof. By extractability, we have $g^{\langle (c_1,...,c_D),(s,....,s^D)\rangle} = T$. Therefore checking the conditions $g^{f(r_1,...,r_\ell)-z} = T$ and $f(r_1, ..., r_\ell) = z + \sum_{j=1}^{D} c_i s^i \mod p$ are equivalent. □

Lemma 5.7. $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_2]$.

Proof. By correctness of Π_{HA}, we have $f(r_1, ..., r_\ell) = \sum_{j=0}^{D} \hat{c}_i s^i \mod p$. (Recall that we defined $\hat{c}_0 := f(x_1, ..., x_\ell)$.) Therefore checking the conditions $f(r_1, ..., r_\ell) = \sum_{j=0}^{D} c_i s^i \mod p$ (where $c_0 := z$) and $Z := \sum_{j=0}^{D}(c_j - \hat{c}_j)s^j \mod p = 0$ are equivalent. Especially, when we have $(c_0, ..., c_D) = (\hat{c}_0, ..., \hat{c}_D)$, this is always satisfied. □

Lemma 5.8. $\Pr[\mathsf{T}_2] = \Pr[\mathsf{T}_3]$.

Proof. Since we set $\sigma_i = (r_i - x_i)s^{-1} \mod p$ in Game_2, and r_i is generated randomly from $\mathbb{Z}_p$, the distribution of σ_i is uniform on $\mathbb{Z}_p$. Moreover, in Game_2, r_i is only used for generating σ_i, and not used by $\mathsf{HA.VerifyEvaled}_2$ at all. Thus σ_i is independent from s, and can be replaced with uniformly random value. □

Lemma 5.9. $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_4]| \leq \frac{(Q+1)D}{p-1}$ *where Q denotes the number of verification queries.*

Proof. Here, we regard the final output $(f^*, x^*, \mathsf{ct}^*)$ of $\mathcal{A}$ as the $(Q+1)$-th query for notational convenience. We consider hybrids H_k for $k = 0, 1, ..., Q+1$, which is the same as Game_3 except that $\mathsf{HA.VerifyEvaled}_4$ is used until $\mathcal{A}$'s k-th query and $\mathsf{HA.VerifyEvaled}_2$ is used for the rest of the queries. Let T'_k be the event that $\mathcal{A}$ wins in H_k. It is clear that H_0 is Game_3 and H_{Q+1} is Game_4. Thus what we have to prove is that we have $|\Pr[\mathsf{T}'_k] - \Pr[\mathsf{T}'_{k+1}]| \leq \frac{D}{p-1}$ for $k = 0, ..., Q$. Let Bad_k be the event that $\mathcal{A}$'s k-th query causes the difference between Game_3 and Game_4, i.e., $(c_0, ..., c_D) \neq (\hat{c}_0, ..., \hat{c}_D)$ and $Z = \sum_{j=0}^{D}(c_j - \hat{c}_j)s^j \mod p = 0$. Since H_k and H_{k+1} are completely the same games as long as Bad_{k+1} does not occur, we have $|\Pr[\mathsf{T}'_k] - \Pr[\mathsf{T}'_{k+1}]| \leq \Pr[\mathsf{Bad}_{k+1}]$. If $(c_0, ..., c_D) \neq (\hat{c}_0, ..., \hat{c}_D)$, then $\sum_{j=0}^{D}(c_j - \hat{c}_j)s^j$ is a non-zero polynomial in s of degree at most D, and have at most D roots. Moreover, in H_k and H_{k+1}, no information of s is used before $\mathcal{A}$ makes its $k+1$-th query since s is not used for generating σ_i due to the modification made in Game_3, and $\mathsf{HA.VerifyEvaled}_4$ does not use s at all. This means that s is uniformly distributed on $\mathbb{Z}_p^*$ from the view of $\mathcal{A}$. Therefore regardless of $\mathcal{A}$'s k-th query, the probability that $\sum_{j=0}^{D}(c_j - \hat{c}_j)s^j = 0$ is at most $\frac{D}{p-1}$. □

Lemma 5.10. $\Pr[\mathsf{T}_4] = 0$.

Proof. Since $\mathsf{HA.VerifyEvaled}_4$ returns $\perp$ if $z \neq f(x_1, ..., x_\ell)$, it is impossible to win this game. □

Combining the above lemmas, we have $\Pr[\mathsf{T}_0] \leq \frac{(Q+1)D}{p-1}$ and thus Π_{HA} is statistically unforgeable. □

Proof of Theorem 5.5

Proof. First, we describe the simulator $\mathsf{HA.Sim}$.

$\mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f, z)$**:** On input vk, ek, sk, f, x, the simulator parses $(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell)) \leftarrow \mathsf{vk}$. Then it sets a vector $(\hat{c}_1, ..., \hat{c}_\ell) := ((f(r_1, ..., r_\ell) - z)s^{-1} \mod p, 0, ..., 0)$. It computes $\mathsf{ct} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{pp}_{\mathsf{IPFE}}, (\hat{c}_1, ..., \hat{c}_\ell))$ and outputs ct.

Suppose that there exists an adversary $\mathcal{A}$ that breaks context-hiding of Π_{HA} w.r.t. the simulator $\mathsf{HA.Sim}$. We construct an adversary $\mathcal{B}$ that breaks the adaptive security of Π_{IPFE} with dimension D based on $\mathcal{A}$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ is given 1^κ and queries the vector dimension 1^D. Then given a public parameter $\mathsf{pp}_{\mathsf{IPFE}}$, it gives 1^κ to $\mathcal{A}$ to obtain message length 1^ℓ, picks $(r_1, ..., r_\ell) \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $s \xleftarrow{\$} \mathbb{Z}_p$, and queries $(s, s^2, ..., s^D)$ to the key generation oracle to obtain $\mathsf{sk}_{\mathsf{IPFE}}$. Then it sets $\mathsf{vk} := (\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}, s, (r_1, ..., r_\ell))$, $\mathsf{ek} := \mathsf{pp}_{\mathsf{IPFE}}$ and $\mathsf{sk} := \mathsf{vk}$. Then it gives $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$ to $\mathcal{A}$, and

$\mathcal{A}$ makes a signing query $(f, \mathbf{x}, \boldsymbol{\sigma})$. It returns $\perp$ if $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \mathbf{x}, \boldsymbol{\sigma}) = \perp$. Otherwise, it computes $(c_1, ..., c_D) \leftarrow \mathsf{Coefficient}(1^D, p, f, \mathbf{x}, \boldsymbol{\sigma})$. It also computes $(\hat{c}_1, ..., \hat{c}_D) := ((f(r_1, ..., r_\ell) - f(x_1, ..., x_\ell))s^{-1} \mod p, 0, ..., 0)$. Then it queries $(c_1, ..., c_D)$ and $(\hat{c}_1, ..., \hat{c}_D)$ as its challenge messages. Given a challenge ciphertext ct, it gives ct to $\mathcal{A}$ as an evaluated signature. Finally, $\mathcal{B}$ outputs as $\mathcal{A}$ outputs.

The above completes the description of $\mathcal{B}$. First, we can see that $\mathcal{B}$ is admissible because $\langle(\hat{c}_1, ..., \hat{c}_n), (s, s^2, ..., s^D)\rangle = f(r_1, ..., r_\ell) - f(x_1, ..., x_\ell) = \langle(c_1, ..., c_n), (s, s^2, ..., s^D)\rangle$ where the first equality follows from $(\hat{c}_1, ..., \hat{c}_n) = ((f(r_1, ..., r_\ell) - f(x_1, ..., x_\ell))s^{-1} \mod p, 0, ..., 0)$ and the second equality follows from the correctness of Π_{HA}. It is easy to see that ct is generated by $\mathsf{HA.Eval}(\mathsf{ek}_{\mathsf{IPFE}}, \mathbf{x}, \boldsymbol{\sigma})$ or $\mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f, f(x_1, ..., x_\ell))$ depending on the coin chosen by the challenger of the adaptive security game of Π_{IPFE} $\mathcal{B}$ is involved. Therefore if $\mathcal{A}$ breaks the context-hiding of Π_{HA} w.r.t. $\mathsf{HA.Sim}$, then $\mathcal{B}$ breaks the adaptive security of Π_{IPFE}. □

By combining the above theorems with Theorem 5.2, we have the following corollary.

Corollary 5.11. *If the DDH assumption holds, then there exists a HomMAC for polynomially bounded degree polynomials over $\mathbb{Z}_p$ with statistical unforgeability and computational context-hiding.*

Remark 5.12. (Garbled-circuit-based FE do not suffice.) The proof of context-hiding only use one-key security of the underlying expIPFE. Then one may think that we can use one-key functional encryption schemes based on garbled circuits [SS10, GVW12] to construct context-hiding HomMAC. However, this is not the case since they do not have extractability, and thus we cannot prove unforgeability. Indeed, it is easy to break the unforgeability if we instantiate the above HomMAC based on those functional encryption schemes.

5.3 Compact PP-NIZK

Using the transformation by Kim and Wu [KW18a] (which is described in Appendix C.2) to our HomMAC from the DDH, we obtain PP-NIZK from the same assumption. Since our HomMAC is not compact, a simple adaptation of their transformation yields PP-NIZK with proof size $O(|C|\kappa) + \mathsf{poly}(\kappa)$. However, by taking advantage of the fact our scheme can deal with arithmetic circuits over $\mathbb{Z}_p$ of polynomial degree, which is larger than $\mathbf{NC}^1$, and incorporating the technique by Katsumata [Kat17], we can reduce the proof size to $|C| + \mathsf{poly}(\kappa)$. See Appendix C.3 for details. Then we obtain the following theorem.

Theorem 5.13. *If the DDH assumption holds on a pairing free group, then there exists PP-NIZK for all **NP** languages with proof size $|C| + \mathsf{poly}(\kappa)$, where $|C|$ denotes the size of circuit that computes the relation being proved.*

Similarly to the case in Section 4, if the proven **NP** relation can be expressed as a leveled circuit, we can further reduce the proof size to $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$. See Appendix C.4 for the details.

Acknowledgement

We would like to thank Geoffroy Couteau for helpful comments on related works and anonymous reviewers of Eurocrypt 2019 for their valuable comments. The first author was partially supported by JST CREST Grant Number JPMJCR1302 and JSPS KAKENHI Grant Number 17J05603. The third author was supported by JST CREST Grant No. JPMJCR1688 and JSPS KAKENHI Grant Number 16K16068.

References

[AB09] Shweta Agrawal and Dan Boneh. Homomorphic MACs: MAC-based integrity for network coding. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 292–305. Springer, Heidelberg, June 2009. (Cited on page 10.)

[ABC+07] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Song. Provable data possession at untrusted stores. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 598–609. ACM Press, October 2007. (Cited on page 10.)

[Abu13] Hamza Abusalah. Generic instantiations of the hidden bits model for non-interactive zero-knowledge proofs for NP, 2013. Master's thesis, RWTH-Aachen University. (Cited on page 3, 4, 5.)

[AC16] Shashank Agrawal and Melissa Chase. A study of pair encodings: Predicate encryption in prime order groups. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 259–288. Springer, Heidelberg, January 2016. (Cited on page 7.)

[AC17] Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, April / May 2017. (Cited on page 7.)

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 575–601. Springer, Heidelberg, November / December 2015. (Cited on page 7, 10.)

[AL11] Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 17–34. Springer, Heidelberg, March 2011. (Cited on page 10.)

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016. (Cited on page 8, 41, 60, 61.)

[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. (Cited on page 3.)

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018. (Cited on page 9.)

[BCH86] Paul W Beame, Stephen A Cook, and H James Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986. (Cited on page 64.)

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Heidelberg, May 2011. (Cited on page 10.)

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16. Springer, Heidelberg, March 2011. (Cited on page 10.)

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Heidelberg, March 2009. (Cited on page 10.)

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. (Cited on page 2.)

[BFR13] Michael Backes, Dario Fiore, and Raphael M. Reischuk. Verifiable delegation of computation on outsourced data. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 863–874. ACM Press, November 2013. (Cited on page 68.)

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 7, 10.)

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016. (Cited on page 4, 8, 9, 65.)

[BGJS16] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. Verifiable functional encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 557–587. Springer, Heidelberg, December 2016. (Cited on page 41.)

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003. (Cited on page 2, 3.)

[BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015. (Cited on page 2.)

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 474–502. Springer, Heidelberg, January 2016. (Cited on page 2.)

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, June 1996. (Cited on page 2.)

[CC18] Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 193–221. Springer, Heidelberg, April / May 2018. (Cited on page 2, 9.)

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. 2019. (Cited on page 9.)

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018. (Cited on page 9.)

[CD04] Ronald Cramer and Ivan Damgård. Secret-key zero-knowlegde and non-interactive verifiable exponentiation. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 223–237. Springer, Heidelberg, February 2004. (Cited on page 3, 9, 12.)

[CDI+18] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. *IACR Cryptology ePrint Archive*, 2018:940, 2018. (Cited on page 2, 9.)

[CF18] Dario Catalano and Dario Fiore. Practical homomorphic message authenticators for arithmetic circuits. *Journal of Cryptology*, 31(1):23–59, January 2018. (Cited on page 3, 4, 7, 10, 15, 43, 67.)

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. (Cited on page 6.)

[CFGN14] Dario Catalano, Dario Fiore, Rosario Gennaro, and Luca Nizzardo. Generalizing homomorphic MACs for arithmetic circuits. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 538–555. Springer, Heidelberg, March 2014. (Cited on page 15.)

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990. (Cited on page 3.)

[CFN15] Dario Catalano, Dario Fiore, and Luca Nizzardo. Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 254–274. Springer, Heidelberg, August 2015. (Cited on page 10.)

[CFN18] Dario Catalano, Dario Fiore, and Luca Nizzardo. On the security notions for homomorphic signatures. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 183–201. Springer, Heidelberg, July 2018. (Cited on page 15.)

[CFW12] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 680–696. Springer, Heidelberg, May 2012. (Cited on page 10.)

[CG15] Pyrros Chaidos and Jens Groth. Making sigma-protocols non-interactive without random oracles. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 650–670. Springer, Heidelberg, March / April 2015. (Cited on page 2, 9.)

[CH19] Geoffroy Couteau and Dennis Hofheinz. Designated-verifier pseudorandom generators, and their applications. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 562–592. Springer, Heidelberg, May 2019. (Cited on page 9.)

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985. (Cited on page 2, 3.)

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007. (Cited on page 3, 4, 5.)

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005. (Cited on page 3.)

[CKS09] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. *Journal of Cryptology*, 22(4):470–504, October 2009. (Cited on page 6, 17.)

[CMP14] Dario Catalano, Antonio Marcedone, and Orazio Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 193–212. Springer, Heidelberg, December 2014. (Cited on page 10.)

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991. (Cited on page 2, 3.)

[Dam90] Ivan Damgård. On the randomness of legendre and jacobi sequences. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 163–172. Springer, Heidelberg, August 1990. (Cited on page 2, 3.)

[Dam93] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 341–355. Springer, Heidelberg, May 1993. (Cited on page 3, 9.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. (Cited on page 2.)

[Des93] Yvo Desmedt. Computer security by redefining what a computer is. In *NSPW*, pages 160–166. ACM, 1993. (Cited on page 10.)

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 41–59. Springer, Heidelberg, March 2006. (Cited on page 2, 9, 12.)

[DMP90] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 269–282. Springer, Heidelberg, August 1990. (Cited on page 3, 9.)

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. (Cited on page 4, 5, 6, 18.)

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. (Cited on page 2, 4, 5, 9, 12, 16, 17, 32.)

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 697–714. Springer, Heidelberg, May 2012. (Cited on page 10, 15.)

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. (Cited on page 2, 9.)

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. (Cited on page 64.)

[GGI+15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, October 2015. (Cited on page 2, 4.)

[GKKR10] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 142–160. Springer, Heidelberg, May 2010. (Cited on page 10.)

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989. (Cited on page 16.)

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. (Cited on page 2.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Cited on page 2, 3.)

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. (Cited on page 2.)

[Gol04] Oded Goldreich. Foundations of cryptography: Volume 2, basic applications. 2004. (Cited on page 2, 4.)

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012. (Cited on page 2, 3, 4.)

[GPSW06a] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 7, 10.)

[GPSW06b] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. *IACR Cryptology ePrint Archive*, 2006:309, 2006. Version 20061007:061901. (Cited on page 34.)

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. (Cited on page 2, 4.)

[GS12] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012. (Cited on page 2.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 8, 45.)

[GVW15a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *J. ACM*, 62(6):45:1–45:33, 2015. (Cited on page 7.)

[GVW15b] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015. (Cited on page 2, 10, 15, 67, 68.)

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320. Springer, Heidelberg, December 2013. (Cited on page 10, 15.)

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th FOCS*, pages 850–858. IEEE Computer Society Press, October 2018. (Cited on page 9.)

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009. (Cited on page 3, 9.)

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, Heidelberg, February 2002. (Cited on page 10.)

[Kat17] Shuichi Katsumata. On the untapped potential of encoding predicates by arithmetic circuits and their applications. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 95–125. Springer, Heidelberg, December 2017. (Cited on page 8, 45, 65.)

[KMO90] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 545–546. Springer, Heidelberg, August 1990. (Cited on page 3, 9.)

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017. (Cited on page 9.)

[KW18a] Sam Kim and David J. Wu. Multi-theorem preprocessing NIZKs from lattices. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 733–765. Springer, Heidelberg, August 2018. (Cited on page 2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 33, 40, 45, 61, 64, 67.)

[KW18b] Sam Kim and David J Wu. Multi-theorem preprocessing nizks from lattices. Cryptology ePrint Archive, Report 2018/272, 2018. https://eprint.iacr.org/2018/272.pdf, Version 20180606:204702. Preliminary version appeared in CRYPTO 2018. (Cited on page 4, 12.)

[Lip17] Helger Lipmaa. Optimally sound sigma protocols under DCRA. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 182–203. Springer, Heidelberg, April 2017. (Cited on page 2, 9.)

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 353–365. Springer, Heidelberg, August 1991. (Cited on page 3, 9.)

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, Heidelberg, May 2011. (Cited on page 34.)

[NP15] Mridul Nandi and Tapas Pandit. On the power of pair encodings: Frameworks for predicate cryptographic primitives. Cryptology ePrint Archive, Report 2015/955, 2015. http://eprint.iacr.org/2015/955. (Cited on page 10.)

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346. Springer, Heidelberg, May 1999. (Cited on page 64.)

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. (Cited on page 64.)

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. (Cited on page 2.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. (Cited on page 2.)

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. *IACR Cryptology ePrint Archive*, 2019:158, 2019. (Cited on page 9.)

[PsV06] Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 271–289. Springer, Heidelberg, August 2006. (Cited on page 2, 9, 12.)

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 593–621. Springer, Heidelberg, May 2019. (Cited on page 9.)

[RSS18] Ron D. Rothblum, Adam Sealfon, and Katerina Sotiraki. Towards non-interactive zero-knowledge for NP from LWE. Cryptology ePrint Archive, Report 2018/240, 2018. https://eprint.iacr.org/2018/240. (Cited on page 2.)

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001. (Cited on page 2.)

[RW12] Yannis Rouselakis and Brent Waters. New constructions and proof methods for large universe attribute-based encryption. *IACR Cryptology ePrint Archive*, 2012:583, 2012. Version 20140828:060226. (Cited on page 34.)

[RW13] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 463–474. ACM Press, November 2013. (Cited on page 7, 34.)

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. (Cited on page 2.)

[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. (Cited on page 34.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. (Cited on page 8, 45.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. (Cited on page 7.)

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 90–107. Springer, Heidelberg, December 2008. (Cited on page 10.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. (Cited on page 2, 4.)

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times anonymous authentication (extended abstract). In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 308–322. Springer, Heidelberg, December 2004. (Cited on page 3.)

[Tsa17] Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 489–518. Springer, Heidelberg, November 2017. (Cited on page 10.)

[VV09] Carmine Ventre and Ivan Visconti. Co-sound zero-knowledge with public keys. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 287–304. Springer, Heidelberg, June 2009. (Cited on page 2, 9.)

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005. (Cited on page 72.)

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, Heidelberg, March 2011. (Cited on page 7.)

A Simpler Variant of DV-NIZK from CDH with Non-Adaptive Zero-Knowledge

Here, we present a simpler variant of DV-NIZK from the CDH assumption given in Section 3 that only achieves *non-adaptive* zero-knowledge where an adversary has to declare statements on which he sees proofs before seeing a common reference string and verification key. Since we can generically convert any DV-NIZK proof system with non-adaptive *single-theorem* zero-knowledge to one with non-adaptive *multi-theorem* zero-knowledge by a standard techbnique similarly to the adaptive case given in Section 3.3, we focus on constructing a DV-NIZK with non-adaptive single-theorem zero-knowledge here.

Definition of Non-Adaptive Single-Theorem Zero-Knowledge. For clarity, we formally define non-adaptive single-theorem zero-knowledge.

Definition A.1. *We say that a DV-NIZK proof system* $\Pi_{\mathsf{DVNIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *satisfies the non-adaptive single-theorem zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *that satisfies the following. For all (stateful) PPT adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{zk\text{-}real}}_{\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{zk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(1^\kappa) = 1]\right| = \mathsf{negl}(\kappa),$$

where experiments $\mathsf{Expt}^{\mathsf{zk\text{-}real}}_{\mathcal{A}}$ *and* $\mathsf{Expt}^{\mathsf{zk\text{-}sim}}_{\mathcal{A},\mathcal{S}}$ *are defined as follows.*

$\mathsf{Expt}^{\mathsf{zk\text{-}real}}_{\mathcal{A}}$
$(x, w) \leftarrow \mathcal{A}(1^\kappa)$,
If $(x, w) \in \mathcal{R}$,
  $(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(1^\kappa)$,
  $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$,
Else $(\mathsf{crs}, k_{\mathsf{V}}, \pi) := (\bot, \bot, \bot)$,
$b' \leftarrow \mathcal{A}(\mathsf{crs}, k_{\mathsf{V}}, \pi)$
outputs b'

$\mathsf{Expt}^{\mathsf{zk\text{-}sim}}_{\mathcal{A},\mathcal{S}}$
$(x, w) \leftarrow \mathcal{A}(1^\kappa)$,
If $(x, w) \in \mathcal{R}$,
  $(\mathsf{crs}, k_{\mathsf{V}}, \pi) \leftarrow \mathcal{S}(1^\kappa, x)$,

Else $(\mathsf{crs}, k_{\mathsf{V}}, \pi) := (\bot, \bot, \bot)$,
$b' \leftarrow \mathcal{A}(\mathsf{crs}, k_{\mathsf{V}}, \pi)$
outputs b'

Construction. Here, we give a construction of a DV-NIZK with non-adaptive single-theorem zero-knowledge. As in Section 3, $\mathsf{GGen}(1^\kappa)$ is an algorithm that generates a group description $(\mathbb{G}, p, g)$ where $2^{2\kappa} \leq p$, and u denotes the length of the binary representation of an element of $\mathbb{G}$. Let $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$ be a NIZK proof system in the HBM with hidden-random-string-length $\ell_{\mathsf{hrs}}(\kappa)$ and soundness error $\epsilon_{\mathsf{HBM}} \leq 2^{-\kappa} \cdot p^{-1}$. We note that this is possible since we can make the soundness error arbitrarily small by increasing $\ell_{\mathsf{hrs}}(\kappa)$ by Theorem 3.7. Hereafter, we simply write ℓ_{hrs} instead of $\ell_{\mathsf{hrs}}(\kappa)$ for ease of notation. The description of our DV-NIZK proof system $\Pi_{\mathsf{DVNIZK}} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is as follows.

$\mathsf{Setup}(1^\kappa)$**:** This algorithm generates the following parameters.

1. Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
2. Samples $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}]$ and a common reference string $\overline{\mathsf{crs}} := \{X_i\}_{i\in[\ell_{\mathsf{hrs}}]} \xleftarrow{\$} \mathbb{G}^{\ell_{\mathsf{hrs}}}$ uniformly at random.
3. Sets $\widehat{\mathsf{crs}} := \{\widehat{X}_i\}_{i\in[\ell_{\mathsf{hrs}}]} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i\in[\ell_{\mathsf{hrs}}]}$.
4. Samples $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$.
5. Outputs a common reference string $\mathsf{crs} := (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$.

We can interpret crs as $(\{X_i, \widehat{X}_i, R_i\}_{i\in[\ell_{\mathsf{hrs}}]}) \in \mathbb{G}^{2\ell_{\mathsf{hrs}}} \times \{0,1\}^{\ell_{\mathsf{hrs}}u}$.

$\mathsf{Prove}(\mathsf{crs}, x, w)$**:** This algorithm does the following.

1. Parses $\mathsf{crs} = (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ where $\overline{\mathsf{crs}} = \{X_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, $\widehat{\mathsf{crs}} = \{\widehat{X}_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, and $\overline{R} = \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$.
2. Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
3. Sets $Z_i := X_i^\tau$ and $\widehat{Z}_i := \widehat{X}_i^\tau$ and $\rho_i = \mathsf{GL}(Z_i; R_i)$ for $i \in [\ell_{\mathsf{hrs}}]$.
4. Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$ where $\rho := \rho_1\| \cdots \|\rho_{\ell_{\mathsf{hrs}}}$.
5. Outputs a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.

$\mathsf{Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi)$**:** This algorithm parses $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_{,i})\}_{i\in I}, T)$, $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$, $\mathsf{crs} = (\mathbb{G}, p, g)\|\overline{\mathsf{crs}}\|\widehat{\mathsf{crs}}\|\overline{R}$ where $\overline{\mathsf{crs}} = \{X_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, $\widehat{\mathsf{crs}} = \{\widehat{X}_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, $\overline{R} = \{R_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, and $s \in \{0,1\}^{\ell_{\mathsf{hrs}}}$. This algorithm does the following.

- For all $i \in I$,

1. Verifies that $\mathsf{Test}_{\mathsf{TDH}}((\alpha_i, \beta_i), X_i, \widehat{X}_i, T, Z_i, \widehat{Z}_i) = \top$, where $\mathsf{Test}_{\mathsf{TDH}}$ is defined in Figure 2 (we present it again for convenience though it is the same as Figure 1). If this equation does not hold, then the verification algorithm immediately outputs $\perp$.
2. Computes $\rho_i = \mathsf{GL}(Z_i; R_i)$.

- If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi_{\mathsf{hbm}}, I, \rho_{|I})$.

The trapdoor test $\mathsf{Test}_{\mathsf{TDH}}((\alpha, \beta), X, \widehat{X}, Y, Z, \widehat{Z})$

1. Verifies that $Z^\alpha \cdot \widehat{Z} = Y^\beta$. If it holds, then outputs $\top$, else $\perp$.

Figure 2: The algorithm $\mathsf{Test}_{\mathsf{TDH}}((\alpha, \beta), X, \widehat{X}, Y, Z, \widehat{Z})$ verifies that $Z = Y^x$ and $\widehat{Z} = Y^{\widehat{x}}$, that is (g, Y, X, Z) and $(g, Y, \widehat{X}, \widehat{Z})$ where $X = g^x$ and $\widehat{X} = g^{\widehat{x}}$ are DDH-tuples without $(x, \widehat{x})$.

Security of Π_{DVNIZK}. Now, we prove the security of the base proof system.

Lemma A.2 (Correctness). *Our base proof system Π_{DVNIZK} satisfies the correctness.*

Proof of Lemma A.2. If $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^\tau)$ is an honestly generated proof, then we have $Z_i = X_i^\tau$ and $\widehat{Z}_i = \widehat{X}_i^\tau$ for $i \in I$. Therefore, an honestly generated proof passes the twin-DH trapdoor test $\mathsf{Test}_{\mathsf{TDH}}$ and a verifier obtains the valid hidden bits ρ via $\mathsf{GL}(Z_i; R_i)$. Thus, we can use the correctness of HBM and the correctness of Π_{DVNIZK} follows. □

Lemma A.3 (Soundness). *If the soundness error of HBM is at most $2^{-\kappa} \cdot p^{-1}$ and the number of verification queries is at most q_v, then Π_{DVNIZK} then Π_{DVNIZK} satisfies the soundness.*

Proof of Lemma A.3. We define a sequence of hybrid games.

Game_0**:** This game is the original experiment of the soundness. Specifically, the game is described as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(1^\kappa)$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Samples $\overline{\mathsf{crs}} := \{X_i\}_{i \in [\ell_{\mathsf{hrs}}]} \xleftarrow{\$} \mathbb{G}^{\ell_{\mathsf{hrs}}}$.
   - Samples $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_i := X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i \in [\ell_{\mathsf{hrs}}]}$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs})$.
3. The experiment plays the role of the verification oracle when $\mathcal{A}$ sends (x', π') as a verification query or the final output as follows.
   - Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i \in I'}, T')$.
   - For all $i \in I'$
     (a) Verifies that $\mathsf{Test}_{\mathsf{TDH}}((\alpha_i, \beta_i), T', Z'_i, \widehat{Z}'_i) = 1$. If this equation does not hold, then the verification oracle immediately returns $\perp$.
     (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
   - If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.
4. $\mathcal{A}$ outputs (x^*, π^*). If $x^* \notin \mathcal{L}$ and π^* passes the verification, then the game outputs 1.

Game_1**:** This game is the same as Game_0 except that it changes how to generate crs and verify proofs as follows.

1. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$ and sets $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Samples $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\overline{\mathsf{crs}} := \{X_i := g^{\chi_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$.
- Samples $\hat{\chi}_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_i := g^{\hat{\chi}_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$.

2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, s)$.
3. When $\mathcal{A}$ sends (x', π') as a verification query or the final output, the experiment does the following.
   - Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i \in I'}, T')$.
   - For all $i \in I'$
     (a) Verifies that $Z'_i = (T')^{\chi_i}$ and $\widehat{Z}'_i = (T')^{\hat{\chi}_i}$ for all $i \in [\ell_{\mathsf{hrs}}]$. If this equation does not hold, then the verification oracle immediately returns $\perp$.
     (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
   - If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.
4. $\mathcal{A}$ outputs (x^*, π^*). If $x^* \notin \mathcal{L}$ and π^* passes the modified verification procedure defined above, then the game outputs 1.

Game_2**:** This game is the same as Game_1 except that the game guesses $\tau \in \mathbb{Z}_p$ at the CRS generation phase and aborts when the guess is wrong. That is, the game first randomly guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$. When $\mathcal{A}$ outputs $(x^*, \pi^* = (\pi^*_{\mathsf{hbm}}, I^*, \{(Z^*_i, \widehat{Z}^*_i)\}_{i \in I'}, T^*)$ as its final output, if $T^* \neq g^\tau$, then this game aborts. Otherwise it works similarly to Game_1.

Game_3**:** This game is the same as Game_2 except that it changes how to generate crs as follows.

1. Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
2. Guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$.
3. Chooses $\rho_i \xleftarrow{\$} \{0,1\}$ for $i \in [\ell_{\mathsf{hrs}}]$.
4. Chooses χ_i and R_i such that $\mathsf{GL}((g^\tau)^{\chi_i}; R_i) = \rho_i$ and sets $X_i := g^{\chi_i}$ for $i \in [\ell_{\mathsf{hrs}}]$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
5. Sets $\overline{\mathsf{crs}} := \{X_i\}_{i \in [\ell_{\mathsf{hrs}}]}$.
6. Samples $\hat{\chi}_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_i := g^{\hat{\chi}_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$.
7. Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$.

We will prove indistinguishability of hybrid games below. We will prove indistinguishability of hybrid games below.

Claim A.4. It holds that $\Pr[\mathsf{Game}_0 = 1] - \Pr[\mathsf{Game}_1 = 1] \leq (q_v + 1)/p$ where q_v is the number of $\mathcal{A}$'s verification queries.

Proof. It can be seen that the distribution of $\{X_i, \widehat{X}_i\}_{i \in [\ell_{\mathsf{hrs}}], b \in \{0,1\}}$ in Game_1 is the same as that in Game_0. We can apply Theorem 3.3 and the claim follows since $\mathcal{A}$ sends at most q_v queries to the verification oracle and the experiment uses the verification oracle once more for deciding if $\mathcal{A}$ succeeds in forging a proof. □

Claim A.5. It holds that $\Pr[\mathsf{Game}_2 = 1] = \frac{1}{p} \Pr[\mathsf{Game}_1 = 1]$.

Proof. The difference between the two games is that the game guesses $\tau \in \mathbb{Z}_p$ and aborts if the guess is incorrect in Game_2. Since the probability that the guess is correct is $1/p$, the claim follows. □

Claim A.6. It holds that $\Pr[\mathsf{Game}_2 = 1] = \Pr[\mathsf{Game}_3 = 1]$.

Proof. In Game_2, ρ_i is computed by $\mathsf{GL}(g^{\tau\chi_i}; R_i)$ using the CRS and τ. On the other hand, in Game_3, we sample $\rho_i \xleftarrow{\$} \{0,1\}$. After that, we choose random χ_i and R_i such that $\rho_i = \mathsf{GL}((g^\tau)^{\chi_i}; R_i)$. (We can assume that any group element in $\mathbb{G}$ is not encoded into the all zero string.) These two distributions are identically distributed since $\rho_i = \mathsf{GL}(g^{\tau\chi_i}; R_i)$ is uniformly random over $\{0,1\}$ when $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ and $R_i \xleftarrow{\$} \{0,1\}^u$ are uniformly and randomly chosen. □

Claim A.7. If the soundness error of HBM is ϵ_{HBM}, then it holds that $\Pr[\mathsf{Game}_3 = 1] \leq \epsilon_{\mathsf{HBM}}$.

Proof. We show that if $\mathcal{A}$ can generate a pair of statement and proof (x^*, π^*) that passes the verification in Game_3, then we can construct a cheating prover $\mathcal{B}$ of $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$. To use $\mathcal{A}$, $\mathcal{B}$ proceeds as follows.

CRS simulation: First, $\mathcal{B}$ is given $(1^\kappa, \rho)$. $\mathcal{B}$ simulates crs as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.
- Guesses $\tau \xleftarrow{\$} \mathbb{Z}_p$.
- Chooses χ_i and R_i such that $\mathsf{GL}((g^\tau)^{\chi_i}; R_i) = \rho_i$ and sets $X_i := g^{\chi_i}$ for $i \in [\ell_{\mathsf{hrs}}]$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\overline{\mathsf{crs}} := \{X_i\}_{i \in [\ell_{\mathsf{hrs}}]}$.
- Samples $\hat{\chi}_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\widehat{\mathsf{crs}} := \{\widehat{X}_i := g^{\hat{\chi}_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and gives crs and s to $\mathcal{A}$.

Verification oracle simulation: When $\mathcal{A}$ sends (x', π') as a verification query or the final output, the experiment does the following.

- Parses $\pi' = (\pi'_{\mathsf{hbm}}, I', \{(Z'_i, \widehat{Z}'_i)\}_{i \in I'}, T')$.
- For all $i \in I'$
  - (a) Verifies that $Z'_i = (T')^{\chi_i}$ and $\widehat{Z}'_i = (T')^{\hat{\chi}_i}$ for all $i \in [\ell_{\mathsf{hrs}}]$. If this equation does not hold, then it immediately returns $\perp$.
  - (b) Computes $\rho'_i := \mathsf{GL}(Z'_i; R_i)$.
- If the proof passes all the tests above, then this algorithm outputs $\mathsf{HBM.Verify}(1^\kappa, x, \pi'_{\mathsf{hbm}}, I', \rho'_{|I'})$.

Forgery: When $\mathcal{A}$ outputs $(x^*, \pi^* = (\pi^*_{\mathsf{hbm}}, I^*, \{(Z^*_i, \widehat{Z}^*_i)\}_{i \in I}, T^*)$, if $T^* \neq g^\tau$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs $(\pi^*_{\mathsf{hbm}}, I^*, \rho_{|I^*})$.

The simulations of $(\mathsf{crs}, k_{\mathsf{V}})$ and the verification oracle are perfect. If Game_3 outputs 1, we have $x^* \notin \mathcal{L}$, $T^* = g^\tau$ and (x^*, π^*) passes the verification oracle in Game_3, which implies we have $Z^*_i = (T^*)^{\chi_i}$ and $\mathsf{HBM.Verify}(1^\kappa, x, \pi^*_{\mathsf{hbm}}, I^*, \rho^*{}_{|I^*}) = 1$ where $\rho^*_i := \mathsf{GL}(Z^*_i; R_i)$. On the other hand, we can see that we have $\mathsf{GL}((g^\tau)^{\chi_i}; R_i) = \rho_i$ by the way of the CRS simulation. Since we have $T^* = g^\tau$, we have $Z^*_i = (g^\tau)^{\chi_i}$ and thus $\rho^*_i = \rho_i$. Hence we have $\mathsf{HBM.Verify}(1^\kappa, x, \pi^*_{\mathsf{hbm}}, I^*, \rho_{|I^*}) = \top$, which means that $\mathcal{B}$ succeeds in forging a proof of HBM. □

By combining the above claims, we have $\Pr[\mathsf{Game}_0 = 1] \leq (q_v + 1)/p + p \cdot \epsilon_{\mathsf{HBM}}$. Since we have $q_v = \mathsf{poly}(\kappa)$ and we assume $\epsilon_{\mathsf{HBM}} \leq 2^{-\kappa} \cdot p^{-1}$, we have $\Pr[\mathsf{Game}_0 = 1] = \mathsf{negl}(\kappa)$ and thus Lemma A.3 is proven. □

Lemma A.8 (Non-Adaptive Single-Theorem Zero-Knowledge). *If the CDH assumption holds with respect to* GGen*, then* Π_{DVNIZK} *satisfies the non-adaptive single-theorem zero-knowledge.*

Proof of Lemma A.8. First, we describe the simulator $\Pi_{\mathsf{DVNIZK}}.\mathcal{S}$.

$\Pi_{\mathsf{DVNIZK}}.\mathcal{S}(1^\kappa, x)$ generates $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ as follows.

- Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$.

- Chooses $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$ and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$, and sets $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$.
- Chooses $\tau \xleftarrow{\$} \mathbb{Z}_p$.
- Generates $(\pi_{\mathsf{hbm}}, I, \rho_{|I}) \leftarrow \mathsf{HBM}.\mathcal{S}(1^\kappa, x)$.
- Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^\tau; R_i) = \rho_i$ for $i \in I$. We can find such χ_i by the sampling algorithm in Lemma 3.12.
- Sets $X_i := g^{\chi_i}$ for $i \in I$.
- Chooses $X_i \xleftarrow{\$} \mathbb{G}$ for $i \in [\ell_{\mathsf{hrs}}] \setminus I$.
- Sets $\overline{\mathsf{crs}} := \{X_i\}_{i\in[\ell_{\mathsf{hrs}}]}$, $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i\in[\ell_{\mathsf{hrs}}]}$, and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} = (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$.
- Sets $Z_i := X_i^\tau$ and $\widehat{Z}_i := \widehat{X}_i^\tau$ for $i \in I$.
- Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
- Outputs $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$.

For proving that a proof simulated by the above simulator is computationally indistinguishable from a real one, we define a sequence of hybrid games. In the following, we assume that (x, w) output by the adversary $\mathcal{A}$ satisfies $(x, w) \in \mathcal{R}$ where $\mathcal{R}$ is the corresponding relation of $\mathcal{L}$. This can be assumed without loss of generality since $\mathcal{A}$ can check if this holds by himself, and if $(x, w) \notin \mathcal{R}$, then the experiment just returns $\perp$ to $\mathcal{A}$.

Game_0**:** In this game, an adversary $\mathcal{A}$ is given an honestly generated proof. Specifically, the game is described as follows.

1. $\mathcal{A}$ is given 1^κ and outputs (x, w).
2. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Samples $\overline{\mathsf{crs}} := \{X_i\}_{i\in[\ell_{\mathsf{hrs}}]} \xleftarrow{\$} \mathbb{G}^{\ell_{\mathsf{hrs}}}$.
   - Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i\in[\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$.
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Computes $\rho_i := \mathsf{GL}(X_i^\tau; R_i)$ and sets $\rho := \rho_1 \| \cdots \| \rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM}.\mathsf{Prove}(1^\kappa, x, w, \rho)$.
   - Sets $Z_i := X_i^\tau$ and $\widehat{Z}_i := \widehat{X}_i^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_1**:** This game is the same as Game_0 except that ρ_i and X_i are generated in the "reversed" order. Specifically, the game is described as follows.

1. $\mathcal{A}$ is given 1^κ and outputs (x, w).
2. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Chooses $\rho_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}]$.

- Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^{\tau}; R_i) = \rho_i$ for all $i \in [\ell_{\mathsf{hrs}}]$. (We can find such χ_i by the sampling algorithm in Lemma 3.12.)
- Sets $X_i := g^{\chi_i}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
- Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i \in [\ell_{\mathsf{hrs}}]}$.

3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^{\tau})$ as follows.
   - Sets $\rho := \rho_1 \| \cdots \| \rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^{\kappa}, x, w, \rho)$.
   - Sets $Z_i := X_i^{\tau}$ and $\widehat{Z}_i := \widehat{X}_i^{\tau}$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^{\tau})$.
4. $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_2**:** This game is the same as Game_1 except that (π_{hbm}, I) is generated before generating crs, and X_i is generated in a different way. Specifically, the game is described as follows.

1. $\mathcal{A}$ is given 1^{κ} and outputs (x, w).
2. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^{\kappa})$, $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Chooses $\rho_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Sets $\rho := \rho_1 \| \cdots \| \rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^{\kappa}, x, w, \rho)$.
   - Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^{\tau}; R_i) = \rho_i$ for all $i \in I$. (We can find such χ_i by the sampling algorithm in Lemma 3.12.)
   - Sets $X_i := g^{\chi_i}$ for all $i \in I$.
   - Chooses $X_i \xleftarrow{\$} \mathbb{G}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$.
   - Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i \in [\ell_{\mathsf{hrs}}]}$.
3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^{\tau})$ as follows.
   - Sets $Z_i := X_i^{\tau}$ and $\widehat{Z}_i := \widehat{X}_i^{\tau}$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^{\tau})$.
4. $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

Game_3**:** This game is the same as Game_2 except that $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$ is generated by the simulator of the NIZK in the HBM. We note that ρ_i for $i \notin I$ is no longer generated in this game.

1. $\mathcal{A}$ is given 1^{κ} and outputs (x, w).
2. The experiment generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^{\kappa})$, $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Generates $(\pi_{\mathsf{hbm}}, I, \rho_{|I}) \leftarrow \mathsf{HBM}.\mathcal{S}(1^{\kappa}, x)$.
   - Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^{\tau}; R_i) = \rho_i$ for all $i \in I$. (We can find such χ_i by the sampling algorithm in Lemma 3.12.) We note that ρ_i is the corresponding bit of $\rho_{|I}$.

- Sets $X_i := g^{\chi_i}$ for all $i \in I$.
- Chooses $X_i \xleftarrow{\$} \mathbb{G}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$.
- Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i\in[\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
- Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$.

3. The experiment generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Sets $Z_i := X_i^\tau$ and $\widehat{Z}_i := \widehat{X}_i^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ is given to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.

By the definition, Game_3 is completely the same as the experiment simulated by $\Pi_{\mathsf{DVNIZK}}.\mathcal{S}$. We will prove the following claims.

Claim A.9. It holds that $\mathsf{Game}_0 \overset{\mathrm{stat}}{\approx} \mathsf{Game}_1$.

Proof. The difference between two games is how to sample X_i and R_i. In Game_0, X_i is uniformly and randomly chosen. In Game_1, the experiment samples $\rho_i \xleftarrow{\$} \{0,1\}$ and after that X_i is set to g^{χ_i} such that $\mathsf{GL}((g^{\chi_i})^\tau; R_i) = \rho_i$. The indistinguishability between these two games directly follows from Lemma 3.12. □

Claim A.10. If the CDH assumption holds with respect to GGen, then it holds that $\mathsf{Game}_1 \overset{\mathrm{c}}{\approx} \mathsf{Game}_2$.

Proof. The difference between Game_1 and Game_2 is that X_i are generated differently for $i \in [\ell_{\mathsf{hrs}}] \setminus I$. Namely, in Game_1, for each $i \in [\ell_{\mathsf{hrs}}] \setminus I$, X_i is set to be g^{χ_i} where χ_i is uniformly chosen from $\mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^\tau; R_i) = \rho_i$. On the other hand, in Game_2, X_i is just independently and uniformly chosen from $\mathbb{G}$. Then it is immediate to prove the indistinguishability between these two games by using Lemma 3.13. Namely, we show that if $\mathcal{A}$ distinguishes Game_1 and Game_2, then we can construct an adversary $\mathcal{B}$ for the experiment defined in Lemma 3.13.

1. $\mathcal{B}$ is given $(1^\kappa, (\mathbb{G}, p, g), g^x, R_1 \| \cdots \| R_\ell)$ where $\ell := \ell_{\mathsf{hrs}} - |I|$, and runs $\mathcal{A}(1^\kappa)$ to obtain (x, w).
2. $\mathcal{B}$ generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Sets $g^\tau := g^x$.
   - Chooses $\rho_i \xleftarrow{\$} \{0,1\}$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Sets $\rho := \rho_1 \| \cdots \| \rho_{\ell_{\mathsf{hrs}}}$.
   - Generates $(\pi_{\mathsf{hbm}}, I) \leftarrow \mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$.
   - Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^\tau; R_i) = \rho_i$ for all $i \in I$. (We can find such χ_i by the sampling algorithm in Lemma 3.12.)
   - Sets $X_i := g^{\chi_i}$ for all $i \in I$.
   - Sends $(\rho_{i_1}, ..., \rho_{i_\ell})$ to the challenger where i_j is the j-th element of $[\ell_{\mathsf{hrs}}] \setminus I$ in the lexicographical order, and receives $(g^{y_{i_1}}, ..., g^{y_{i_\ell}})$.
   - Sets $X_{i_j} := g^{y_{i_j}}$ for all $j \in [\ell]$. We note that X_i is defined for all $i \in [\ell_{\mathsf{hrs}}]$ at this point.
   - Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i\in[\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i\in[\ell_{\mathsf{hrs}}]}$.
3. $\mathcal{B}$ generates a proof $\pi = (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$ as follows.
   - Sets $Z_i := (g^\tau)^{\chi_i}$ and $\widehat{Z}_i := (g^\tau)^{-\alpha_i\chi_i+\beta_i}$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i\in I}, g^\tau)$.
4. Finally, $\mathcal{B}$ gives $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ to $\mathcal{A}$, and outputs the same as $\mathcal{A}$ outputs.

If $(g^{y_{i_1}}, ..., g^{y_{i_\ell}})$ comes from the experiment $\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{B}}(1^\kappa, 0)$, then $\mathcal{B}$ perfectly simulates Game_2 for $\mathcal{A}$ since they are just uniform and independent group elements. On the other hand, if they come from the experiment $\mathsf{Expt}^{\mathsf{switch}}_{\mathcal{B}}(1^\kappa, 1)$, then the simulation $\mathcal{B}$ perfectly simulates Game_1 for $\mathcal{A}$ since $y_{i_j} \xleftarrow{\$} \mathbb{Z}_p$ is chosen subject to $\mathsf{GL}(g^{xy_{i_j}}; R_{i_j}) = \rho_{i_j}$ for $j \in [\ell]$. Therefore, $\mathcal{B}$ can break the CDH assumption by using $\mathcal{A}$ that distinguishes these two hybrid games. This completes the proof of the claim. □

Claim A.11. If $(\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$ is a NIZK proof system in the HBM, then $\mathsf{Game}_2 \overset{\mathrm{stat}}{\approx} \mathsf{Game}_3$.

Proof. We construct a distinguisher $\mathcal{D}$ of HBM by using a distinguisher $\mathcal{A}$ of Game_2 and Game_3. $\mathcal{D}$ is given 1^κ, then does the following.

1. $\mathcal{D}$ runs $\mathcal{A}(1^\kappa)$ to obtain (x, w), and queries (x, w) to the challenger to receive $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$.
2. $\mathcal{D}$ generates $(\mathsf{crs}, k_{\mathsf{V}})$ as follows.
   - Samples $(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $(\alpha_i, \beta_i) \xleftarrow{\$} \mathbb{Z}_p^2$, and $R_i \xleftarrow{\$} \{0,1\}^u$ for all $i \in [\ell_{\mathsf{hrs}}]$.
   - Samples $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Chooses $\chi_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\mathsf{GL}((g^{\chi_i})^\tau; R_i) = \rho_i$ for all $i \in I$. (We can find such χ_i by the sampling algorithm in Lemma 3.12.) We note that ρ_i is the corresponding bit of $\rho_{|I}$.
   - Sets $X_i := g^{\chi_i}$ for all $i \in I$.
   - Chooses $X_i \xleftarrow{\$} \mathbb{G}$ for all $i \in [\ell_{\mathsf{hrs}}] \setminus I$.
   - Sets $\widehat{\mathsf{crs}} := \{X_i^{-\alpha_i} \cdot g^{\beta_i}\}_{i \in [\ell_{\mathsf{hrs}}]}$ and $\overline{R} := R_1 \| \cdots \| R_{\ell_{\mathsf{hrs}}}$.
   - Sets $\mathsf{crs} := (\mathbb{G}, p, g) \| \overline{\mathsf{crs}} \| \widehat{\mathsf{crs}} \| \overline{R}$ and a verification key $k_{\mathsf{V}} := \{(\alpha_i, \beta_i)\}_{i \in [\ell_{\mathsf{hrs}}]}$.
3. $\mathcal{D}$ generates a proof π as follows.
   - Sets $Z_i := X_i^\tau$ and $\widehat{Z}_i := \widehat{X}_i^\tau$ for $i \in I$.
   - Sets a proof $\pi := (\pi_{\mathsf{hbm}}, I, \{(Z_i, \widehat{Z}_i)\}_{i \in I}, g^\tau)$.
4. Gives $(\mathsf{crs}, k_{\mathsf{V}}, \pi)$ to $\mathcal{A}$, and $\mathcal{A}$ outputs b'.
5. $\mathcal{D}$ outputs b'.

It is clear that $\mathcal{D}$ perfectly simulates Game_2 if ρ is randomly chosen and then (π_{hbm}, I) is generated by $\mathsf{HBM.Prove}(1^\kappa, x, w, \rho)$, and $\mathcal{D}$ perfectly simulates Game_3 if $(\pi_{\mathsf{hbm}}, I, \rho_{|I})$ is generated by $\mathsf{HBM}.\mathcal{S}(1^\kappa, x)$. This completes the proof. □

By the above claims, we complete the proof of Lemma A.8 □

B Omitted Contents of HomMAC from Inner Product Functional Encryption

B.1 Proof of Theorem 5.2

Proof. Agrawal et al. [ALS16] constructed an adaptively secure IPFE scheme based on the DDH assumption. It is easy to see that their construction can be seen as an expIPFE scheme that satisfies extractability. For completeness, we describe the scheme below.

Instantiation. Let GGen be a group generator that generates a description of a group $\mathbb{G}$ along with its order p and its generator g. The expIPFE scheme $\Pi_{\mathsf{ALS}} = (\mathsf{Setup}_{\mathsf{ALS}}, \mathsf{KeyGen}_{\mathsf{ALS}}, \mathsf{Enc}_{\mathsf{ALS}}, \mathsf{Dec}_{\mathsf{ALS}})$ is described as follows.

$\mathsf{Setup}_{\mathsf{ALS}}(1^\kappa, 1^\ell)$**:** It generates $\mathcal{G} = (\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, samples $h \xleftarrow{\$} \mathbb{G}$ and $s_i, t_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [\ell]$, computes $h_i := g^{s_i} \cdot h^{t_i}$ for $i \in [\ell]$, and outputs a public parameter $\mathsf{pp} := (\mathcal{G}, h, \{h_i\}_{i\in[\ell]})$ and a master secret key $\mathsf{msk} := (\mathsf{pp}, \{(s_i, t_i)\}_{i\in[\ell]})$. We denote vectors $(s_1, ..., s_\ell)$ and $(t_1, ..., t_\ell)$ by $\mathbf{s}$ and $\mathbf{t}$ respectively.

$\mathsf{KeyGen}_{\mathsf{ALS}}(\mathsf{msk}, \mathbf{y} = (y_1, ..., y_\ell))$**:** It outputs $\mathsf{sk}_{\mathbf{y}} := (\mathbf{y}, s_{\mathbf{y}}, t_{\mathbf{y}}) := (\mathbf{y}, \langle \mathbf{s}, \mathbf{y}\rangle, \langle \mathbf{t}, \mathbf{y}\rangle)$.

$\mathsf{Enc}(\mathsf{pp}, \mathbf{x} = (x_1, ..., x_\ell))$**:** It samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $C := g^r$, $D := h^r$, $E_i := g^{x_i} \cdot h_i^r$ for $i \in [\ell]$, and outputs a ciphertext $\mathsf{ct}_{\mathbf{x}} := (C, D, \{E_i\}_{i\in[\ell]})$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\mathbf{y}})$ **:** If $\mathsf{ct}_{\mathbf{x}} \notin \mathbb{G}^{\ell+2}$, it outputs $\perp$. Otherwise it parses $(C, D, \{E_i\}_{i\in[\ell]}) \leftarrow \mathsf{ct}_{\mathbf{x}}$ and $(\mathbf{y} = (y_1, ..., y_\ell), s_{\mathbf{y}}, t_{\mathbf{y}}) \leftarrow \mathsf{sk}_{\mathbf{y}}$, and outputs $\mathsf{d} = (\prod_{i=1}^n E_i^{y_i})/(C^{s_{\mathbf{y}}} \cdot D^{t_{\mathbf{y}}})$.

The above scheme is exactly the same as the one proposed by Agrawal et al. [ALS16] except that the decryption algorithm does not compute the discrete logarithm. Therefore correctness and adaptive security of the scheme can be reduced to those of Agrawal et al.'s scheme. Here, we prove that the above scheme is extractable. We describe an unbounded-time extractor Ext below.

$\mathsf{Ext}(\mathsf{msk}, \mathsf{ct})$**:** If $\mathsf{ct} \notin \mathbb{G}^{\ell+2}$, it aborts. Otherwise, it parses $(\mathsf{pp} = (\mathcal{G}, g, h, \{h_i\}_{i\in[\ell]}), \{(s_i, t_i)\}_{i\in[\ell]}) \leftarrow \mathsf{msk}$ and $(C, D, \{E_i\}_{i\in[\ell]}) \leftarrow \mathsf{ct}$. It computes $x_i := \mathsf{Dlog}_g(E_i/(C^{s_i} \cdot D^{t_i}))$ for $i \in [\ell]$ where Dlog_g denotes the discrete logarithm with a base g. Then it outputs $\mathbf{x} := (x_1, ..., x_\ell)$.

We prove that the above extractor works correctly. Let $\mathsf{sk}_{\mathbf{y}}$ be an honestly generated secret key associated with a vector $\mathbf{y}$. Then it is of the form as $(\mathbf{y}, \langle \mathbf{s}, \mathbf{y}\rangle, \langle \mathbf{t}, \mathbf{y}\rangle)$. For any ciphertext $\mathsf{ct} = (C, D, \{E_i\}_{i\in[n]})$, we have $\mathsf{Dec}(\mathsf{pp}, \mathsf{ct}, \mathsf{sk}_{\mathbf{y}}) = (\prod_{i=1}^n E_i^{y_i})/(C^{\langle \mathbf{s},\mathbf{y}\rangle} \cdot D^{\langle \mathbf{t},\mathbf{y}\rangle}) = \prod_{i=1}^n (E_i/(C^{s_i} \cdot D^{t_i}))^{y_i} = \prod_{i=1}^n (g^{x_i})^{y_i} = g^{\langle \mathbf{x},\mathbf{y}\rangle}$ where $\mathbf{x} = (x_1, ..., x_\ell) := \mathsf{Ext}(\mathsf{msk}, \mathsf{ct})$. This concludes the proof of the theorem. □

C PP-NIZK from Homomorphic Authenticators

In this section, we describe a construction of PP-NIZK for bounded **NP** languages based on any context-hiding HomAuth given by Kim and Wu [KW18a]. Since our definitions of HomAuth are slightly different from the ones in [KW18a], we describe the construction and its security proof for completeness. We first prepare the standard definition of symmetric key encryption.

C.1 Preparation: Symmetric Key Encryption

Let $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ be the family of the message space. In the following, we occasionally drop the subscript and simply write $\mathcal{M}$ when the meaning is clear. An symmetric key encryption (SKE) scheme Π_{SKE} is defined by the following three algorithms:

$\mathsf{SKE.KeyGen}(1^\kappa) \to \mathsf{K}_{\mathsf{SKE}}$: The key generation algorithm takes as input the security parameter 1^κ and outputs a secret key $\mathsf{K}_{\mathsf{SKE}}$.

$\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{M}) \to \mathsf{ct}$: The encryption algorithm takes as input a secret key $\mathsf{K}_{\mathsf{SKE}}$ and a message $\mathsf{M} \in \mathcal{M}$ and outputs a ciphertext ct.

$\mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{ct}) \to \mathsf{M}$ or $\perp$: The decryption algorithm takes as input a secret key $\mathsf{K}_{\mathsf{SKE}}$ and a ciphertext ct and outputs a message $\mathsf{M} \in \mathcal{M}$ or a special symbol $\perp$ indicating decryption failure.

Correctness. We require correctness: that is, for all $\kappa \in \mathbb{N}$, $\mathsf{M} \in \mathcal{M}$, and $\mathsf{K}_{\mathsf{SKE}} \in \mathsf{SKE.KeyGen}(1^\kappa)$, we have $\mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{M})) = \mathsf{M}$.

CPA-Security. We say that an SKE scheme satisfies CPA-security if for all $\kappa \in \mathbb{N}$, the following holds: For all PPT adversaries $\mathcal{A}$, if we run $\mathsf{K}_{\mathsf{SKE}} \leftarrow \mathsf{SKE.KeyGen}(1^\kappa)$, then we have

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{K}_{\mathsf{SKE}},\cdot,\cdot)}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\mathsf{K}_{\mathsf{SKE}},\cdot,\cdot)}(1^\kappa) = 1]\right| = \mathsf{negl}(\kappa),$$

where $\mathcal{O}_b(\mathsf{K}_{\mathsf{SKE}}, \mathsf{M}_0, \mathsf{M}_1)$ outputs $\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{M}_b)$ for $b \in \{0,1\}$.

In our construction of preprocessing NIZKs in the following section, we require a symmetric key encryption scheme whose ciphertext overhead (i.e., $|\mathsf{ct}| - |m|$) is $\mathsf{poly}(\kappa)$ and whose decryption algorithm is implemented in $\mathbf{NC}^1$. We use a construction based on the CDH assumption. (See the last paragraph of Appendix C.2 for details.)

C.2 Kim-Wu Construction

Before describing the construction, we prepare some building blocks and notations.

- Let $\mathcal{L}$ be an **NP** language defined by a relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$. Let $n(\kappa)$ and $m(\kappa)$ be any fixed polynomials. Let C be a circuit that computes the relation $\mathcal{R}$ on $\{0,1\}^n \times \{0,1\}^m$, i.e., for $(x,w) \in \{0,1\}^n \times \{0,1\}^m$, we have $C(x,w) = 1$ if and only if $(x,w) \in \mathcal{R}$
- Let $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric key encryption scheme with ciphertext space $\mathcal{CT}$ and key space $\{0,1\}^\ell$.
- For $x \in \{0,1\}^n$ and $\mathsf{ct} \in \mathcal{CT}$, we define the function $f_{x,\mathsf{ct}}(\mathsf{K}_{\mathsf{SKE}}) := C(x, \mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{ct}))$.
- Let $\mathsf{HA} = (\mathsf{HA.KeyGen}, \mathsf{HA.Sign}, \mathsf{HA.Eval}, \mathsf{HA.VerifyFresh}, \mathsf{HA.VerifyEvaled})$ be a HomAuth scheme Π_{HA} that supports a function class that contains $\{f_{x,\mathsf{ct}}\}_{x \in \{0,1\}^n, \mathsf{ct} \in \mathcal{CT}}$.

The PP-NIZK $\Pi_{\mathsf{PPNIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ is described as follows.

$\mathsf{Setup}(1^\kappa)$**:** This algorithm generates $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(1^\kappa)$, $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$, and $\boldsymbol{\sigma} \xleftarrow{\$} \mathsf{HA.Sign}(\mathsf{sk}, \mathsf{K}_{\mathsf{SKE}})$. It outputs a common reference string $\mathsf{crs} = \mathsf{ek}$ a prover key $k_{\mathsf{P}} := (\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$ and a verifier key $k_{\mathsf{V}} := (\mathsf{vk}, \mathsf{sk})$.[10]

$\mathsf{Prove}(\mathsf{crs}, k_{\mathsf{P}}, x, w)$**:** This algorithm aborts if $\mathcal{R}(x,w) = 0$. Otherwise it parses $\mathsf{ek} \leftarrow \mathsf{crs}$, $(\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma}) \leftarrow k_{\mathsf{P}}$, computes $\mathsf{ct} := \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$, generates $\sigma \xleftarrow{\$} \mathsf{HA.Eval}(\mathsf{ek}, f_{x,\mathsf{ct}}, \mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$, and outputs a proof $\pi := (\mathsf{ct}, \sigma)$.

$\mathsf{Verify}(\mathsf{crs}, k_{\mathsf{V}}, x, \pi)$**:** This algorithm parses $\mathsf{ek} \leftarrow \mathsf{crs}$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow k_{\mathsf{V}}$ and $(\mathsf{ct}, \sigma) \leftarrow \pi$, and outputs $\mathsf{Verify}(\mathsf{vk}, f_{x,\mathsf{ct}}, 1, \sigma)$.

Correctness. Let $\mathsf{crs} := \mathsf{ek}$, $k_{\mathsf{P}} := (\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$ and $k_{\mathsf{V}} := (\mathsf{vk}, \mathsf{sk})$ be generated by $\mathsf{Setup}(1^\kappa)$, and $\pi = (\mathsf{ct}, \sigma)$ be a proof generated by $\mathsf{Prove}(k_{\mathsf{P}}, x, w)$ for $(x,w) \in \mathcal{L}$. Then we have $f_{x,\mathsf{ct}}(\mathsf{K}_{\mathsf{SKE}}) = C(x, \mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{ct})) = C(x,w) = 1$. Therefore $\mathsf{Verify}(k_{\mathsf{V}}, x, \pi)$ outputs $\top$ by the correctness of Π_{HA}.
Security. The security of Π_{PPNIZK} is stated as follows.

Theorem C.1. *If Π_{HA} satisfies selective (resp. statistical) unforgeability and computational context-hiding, and SKE is CPA-secure, then Π_{PPNIZK} satisfies computational (resp. statistical) soundness and non-programmable CRS zero-knowledge.*

Proof. Below, we prove soundness and non-programmable CRS zero-knowledge.
Soundness. We first prove soundness. This part relies on the (selective) single-shot unforgeability of Π_{HA}. Suppose that there exists an adversary $\mathcal{A}$ that breaks the soundness of Π_{PPNIZK}. Then we construct an adversary $\mathcal{B}$ that breaks the unforgeability of Π_{HA}.

$\mathcal{B}$ first chooses $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(1^\kappa)$ and queries $\mathsf{K}_{\mathsf{SKE}}$ as its signing query. Then, it is given an evaluation key ek and a signature $\boldsymbol{\sigma}$. It sets $\mathsf{crs} := \mathsf{ek}$ and $k_{\mathsf{P}} := (\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$, and gives $(\mathsf{crs}, k_{\mathsf{P}})$ to $\mathcal{A}$. When $\mathcal{A}$ makes a verification query (x, π), $\mathcal{B}$ parses $(\mathsf{ct}, \sigma) \leftarrow \pi$, queries $(f_{x,\mathsf{ct}}, 1, \sigma)$ to its own verification oracle, and relays the response from the oracle to $\mathcal{A}$. Finally, when $\mathcal{A}$ outputs a forgery $(x^*, \pi^* = (\mathsf{ct}^*, \sigma^*))$, $\mathcal{B}$ outputs $(f_{x^*,\mathsf{ct}^*}, 1, \sigma^*)$ as its forgery.

It is clear that $\mathcal{B}$ perfectly simulates the experiment that defines soundness to $\mathcal{A}$. Moreover, if $\mathcal{A}$ succeeds in breaking the soundness of Π_{PPNIZK}, then $\mathcal{B}$ also succeeds in breaking the unforgeability of Π_{HA} since if $x^* \notin \mathcal{L}$, $C(x^*, \cdot)$ never outputs 1 on any witness, and specifically f_{x^*,ct^*} never outputs 1 on any input. It is easy to see that the reduction also works in the setting where both $\mathcal{A}$ and $\mathcal{B}$ are unbounded-time adversaries.

[10] In the real protocol, sk is not used at all. This is included in k_{V} as an artifact for proving non-programmable CRS zero-knowledge, instead of programmable CRS zero-knowledge.

(Non-Programmable CRS) Zero-knowledge. Next, we prove that Π_{PPNIZK} satisfies non-programmable CRS zero-knowledge. This part relies on the context-hiding of Π_{HA}. Let $\mathsf{HA.Sim}$ be a simulator for context-hiding of Π_{HA}. We describe a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for zero-knowledge of Π_{PPNIZK} as follows.

$\mathcal{S}_1(1^\kappa, \mathsf{crs} = \mathsf{ek}, k_{\mathsf{V}} = (\mathsf{vk}, \mathsf{sk}))$**:** It generates $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(1^\kappa)$ and returns $\tau_{\mathsf{V}} := \mathsf{K}_{\mathsf{SKE}}$.

$\mathcal{S}_2(\mathsf{crs} = \mathsf{ek}, k_{\mathsf{V}} = (\mathsf{vk}, \mathsf{sk}), \tau_{\mathsf{V}} = \mathsf{K}_{\mathsf{SKE}}, x)$**:** It generates $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, 0^m)$, runs $\sigma \xleftarrow{\$} \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f_{x,\mathsf{ct}}, 1)$, and outputs $\pi := (\mathsf{ct}, \sigma)$.

We prove that proofs generated by $\mathcal{S}$ is computationally indistinguishable from honestly generated proofs. To prove this, we consider the following sequence of games between a PPT adversary $\mathcal{A}$ and a challenger.

Game_0**:** In this game, proofs are generated honestly. Namely,

1. The challenger generates $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(1^\kappa)$, $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \xleftarrow{\$} \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$, and $\boldsymbol{\sigma} \xleftarrow{\$} \mathsf{HA.Sign}(\mathsf{sk}, \mathsf{K}_{\mathsf{SKE}})$, and sets $\mathsf{crs} = \mathsf{ek}$, $k_{\mathsf{P}} := (\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$ and $k_{\mathsf{V}} := (\mathsf{vk}, \mathsf{sk})$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$, and allowed to query $\mathcal{O}(\mathsf{crs}, k_{\mathsf{P}}, \cdot, \cdot)$, which works as follows. When $\mathcal{A}$ queries (x, w), if $(x, w) \notin \mathcal{R}$, then the oracle returns $\bot$. Otherwise, it computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$, generates $\sigma \xleftarrow{\$} \mathsf{HA.Eval}(\mathsf{ek}, f_{x,\mathsf{ct}}, \mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$, and returns $\pi := (\mathsf{ct}, \sigma)$.
3. Finally, $\mathcal{A}$ returns a bit β.

Game_1**:** This game is the same as the previous game except that σ is generated as $\mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f_{x,\mathsf{ct}}, 1)$ in each oracle query. We note that $\boldsymbol{\sigma}$ is not needed to be generated since it is not used at all in this game.

Game_2**:** This game is the same as the previous game except that ct is generated as $\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, 0^m)$.

Let T_i be the event that $\mathcal{A}$ returns 1 in Game_i for $i = 0, 1, 2$. It is easy to see that proofs are generated by $\mathsf{HA.Sim}$ in Game_2. Thus we have to prove that $|\Pr[\mathsf{T}_0] - \Pr[\mathsf{T}_2]|$ is negligible. We prove this by the following lemmas.

Lemma C.2. *If Π_{HA} is computationally context-hiding w.r.t. the simulator $\mathsf{HA.Sim}$, then $|\Pr[\mathsf{T}_0] - \Pr[\mathsf{T}_1]| = \mathsf{negl}(\kappa)$.*

Proof. We consider hybrids H_j for $j \in \{0, 1, ..., Q\}$ where Q denotes the number of $\mathcal{A}$'s queries. In the hybrid H_j, the proof oracle for $\mathcal{A}$ works as in Game_1 for the first j queries, and as in Game_0 for the rest of queries. It is easy to see that H_0 is exactly the same as Game_0, and H_Q is exactly the same as Game_1. Let T'_j be the event that $\mathcal{A}$ outputs 1 in H_j for $j \in \{0, 1, ..., Q\}$. Since we have $|\Pr[\mathsf{T}_0] - \Pr[\mathsf{T}_1]| \leq \sum_{j=1}^{Q} |\Pr[\mathsf{T}'_{j-1}] - \Pr[\mathsf{T}'_j]|$, we only have to prove that $|\Pr[\mathsf{T}'_{j-1}] - \Pr[\mathsf{T}'_j]| = \mathsf{negl}(\kappa)$ for all $j \in [Q]$. Suppose that $|\Pr[\mathsf{T}'_{j-1}] - \Pr[\mathsf{T}'_j]|$ is non-negligible. Then we construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the context-hiding w.r.t. the simulator $\mathsf{HA.Sim}$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ is given $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$, generates $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(1^\kappa)$ and $\boldsymbol{\sigma} \xleftarrow{\$} \mathsf{HA.Sign}(\mathsf{sk}, \mathsf{K}_{\mathsf{SKE}})$, and sets $\mathsf{crs} := \mathsf{ek}$, $k_{\mathsf{P}} := (\mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$, and $k_{\mathsf{V}} := (\mathsf{vk}, \mathsf{sk})$. Then it gives $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$. When $\mathcal{A}$ queries its i-th (x, w), if $(x, w) \notin \mathcal{R}$, $\mathcal{B}$ returns $\bot$. Otherwise it computes as follows.

- If $i \leq j - 1$, then it computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$ and $\sigma \xleftarrow{\$} \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f_{x,\mathsf{ct}}, 1)$, and returns (ct, σ).
- If $i = j$, then it computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$ and queries $(f_{x,\mathsf{ct}}, \mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$ to its own oracle to obtain σ, and returns (ct, σ).
- If $i \geq j + 1$, then it computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$ and $\sigma \xleftarrow{\$} \mathsf{HA.Eval}(\mathsf{ek}, f_{x,\mathsf{ct}}, \mathsf{K}_{\mathsf{SKE}}, \boldsymbol{\sigma})$, and returns (ct, σ).

Finally, when $\mathcal{A}$ outputs β, $\mathcal{B}$ also outputs β.

This completes the description of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ perfectly simulates H_{j-1} or H_j depending on the coin of the challenger of the experiment defining context-hiding in which $\mathcal{B}$ is involved. Thus $|\Pr[\mathsf{T}'_{j-1}] - \Pr[\mathsf{T}'_j]|$ is negligible if Π_{HA} satisfies context-hiding. □

Lemma C.3. *If* SKE *is CPA-secure, then* $|\Pr[\mathsf{T}_1] - \Pr[\mathsf{T}_2]| = \mathsf{negl}(\kappa)$.

Proof. Suppose that $|\Pr[\mathsf{T}_1] - \Pr[\mathsf{T}_2]|$ is non-negligible. Then we construct an adversary $\mathcal{B}$ that breaks the CPA-security of SKE. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ is given 1^κ, generates $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$, sets $\mathsf{crs} := \mathsf{ek}$ and $k_{\mathsf{V}} := (\mathsf{vk}, \mathsf{sk})$, and gives $(1^\kappa, \mathsf{crs}, k_{\mathsf{V}})$ to $\mathcal{A}$ where the prove oracle for $\mathcal{A}$ is simulated as follows. When $\mathcal{A}$ queries (x, w) to the oracle, it returns $\bot$ if $(x, w) \notin \mathcal{R}$. Otherwise, it queries $(0^m, w)$ to its encryption oracle to obtain ct, computes $\sigma \xleftarrow{\$} \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, f_{x,\mathsf{ct}}, 1)$ and returns (ct, σ). Finally, when $\mathcal{A}$ outputs β, $\mathcal{B}$ outputs β.

It is easy to see that $\mathcal{B}$ perfectly simulates Game_1 or Game_2 depending on the random coin chosen by the challenger in the SKE game in which $\mathcal{B}$ is involved. Therefore we have $|\Pr[\mathsf{T}_1] - \Pr[\mathsf{T}_2]| = \mathsf{negl}(\kappa)$ if SKE is CPA-secure. □

This completes the proof of Theorem C.1. □

Remark C.4 (Public verifiability). As observed by Kim and Wu [KW18a], if an underlying HomAuth scheme is publicly verifiable (i.e., HomSig scheme), then the resulting PP-NIZK can also be made publicly verifiable (i.e., DP-NIZK) by letting k_{V} public at the cost of weakening the zero-knowledge property to programmable CRS zero-knowledge from non-programmable CRS zero-knowledge.

Instantiations. Here, we discuss that HomAuth schemes for $\mathbf{NC}^1$ suffice to instantiate the above construction if we assume an additional mild assumption. Specifically, the above construction requires Π_{HA} to support a function class that contains $\{f_{c,\mathsf{ct}}\}_{x\in\{0,1\}^n, \mathsf{ct}\in\mathcal{CT}}$. On the other hand, constructions of HomAuth given in this paper only supports $\mathbf{NC}^1$. Therefore we have to assume that $\{f_{c,\mathsf{ct}}\}_{x\in\{0,1\}^n, \mathsf{ct}\in\mathcal{CT}}$ is contained in $\mathbf{NC}^1$. This is possible if we assume the following.

- $C(x, \cdot)$ is computable in $\mathbf{NC}^1$. This can be assumed without loss of generality by using $\{s_g\}_{g\in\mathsf{Gates}}$ as a witness for the language where Gates is the set of all gates of $C(x, \cdot)$ and s_g is the output value of the gate g when $C(x, \cdot)$ is evaluated on input w. This reduces the depth of a circuit to compute $C(x, \cdot)$ at the cost of expanding the size of a witness to be as large as the size of $C(x, \cdot)$. We note that the same trick was also used in [GGH+16].
- SKE.Dec can be computable in $\mathbf{NC}^1$. Such an SKE scheme can be constructed based on any weak PRF computable in $\mathbf{NC}^1$. We can construct such a weak PRF from the CDH assumption on $\mathbb{Z}_p^*$ for sufficiently large prime p by modifying the weak PRF from the DDH assumption due to Naor et al. [NPR99] so that we generate the output bits using the Goldreich-Levin hardcore function. In order to make the evaluation circuit of the PRF be in $\mathbf{NC}^1$, we introduce a preprocessing on an input similarly to Naor-Reingold PRF [NR04] and change the evaluation circuit so that it computes multiple product (rather than exponentiation) followed by the computation of the Goldreich-Levin hardcore function. The former operation is in $\mathbf{NC}^1$ due to the result by [BCH86] and the latter is actually the inner-product function, which is in $\mathbf{NC}^1$ as well.

Especially, we can instantiate the above construction based on any of the HomAuth schemes constructed in this paper. The proof sizes of the NIZKs obtained by these instantiations are $\mathsf{poly}(\kappa, |C|)$.

C.3 More Compact Proof from HomAuth for Arithmetic Circuits

Here, we show that we can reduce the proof size with an additional trick if we instantiate the scheme based on a HomAuth scheme for arithmetic circuits of polynomial degree over $\mathbb{Z}_p$, which is given in Section 5. Namely, the scheme can be instantiated under the DDH assumption, and its proof size is $|C| + \mathsf{poly}(\kappa)$.

For simplicity, we assume that $C(x, \cdot)$ consists of only NAND gates. (This is just for notational simplicity, and a similar result can be obtained as long as each gate of C can be expressed as a constant degree polynomial over $\mathbb{Z}_p$.) Now, we fix $(x, w) \in \mathcal{R} \cap \{0,1\}^n \times \{0,1\}^m$ and a prime $p > |C|$. Let Gates denote the set of all gates of $C(x, \cdot)$, and for each gate $g \in \mathsf{Gates}$ of the circuit $C(x, \cdot)$ let s_g denote the output value of g when we evaluate $C(x, \cdot)$ on the input w. For each gate g of the circuit C, let A_g and B_g denote the gates whose output wire is connected to the input wire of g. Then we have $s_g = 1 - s_{A_g} \cdot s_{B_g} \mod p$ for every gate g and $s_{\mathsf{out}} = 1 \mod p$ where out is the output gate of $C(x, \cdot)$. Conversely, it is easy to see that there exists $w \in \{0,1\}^m$ such that $C(x, w) = 1$ if and only if there

exists $\{s_g\}_{g\in\mathsf{Gates}} \in \{0,1\}^{|\mathsf{Gates}|}$ such that $s_g = 1 - s_{A_g}\cdot s_{B_g} \mod p$ for every gate g and $s_{\mathsf{out}} = 1 \mod p$, which is equivalent to

$$\prod_{g\in\mathsf{Gates}} (1-(1-s_{A_g}\cdot s_{B_g}-s_g)^2)\cdot(1-(1-s_{\mathsf{out}})^2) = 1 \mod p. \tag{8}$$

Here, we apply a similar trick to the one used by Katsumata [Kat17] to reduce the degree of the above equation. Namely, we first remark that the above equation is satisfied if and only if all terms $(1-s_{A_g}\cdot s_{B_g}-s_g)$ and $(1-s_{\mathsf{out}})$ are 0. Also, remark that $|\mathsf{Gates}| \le |C| < p$ and all terms $(1-s_{A_g}\cdot s_{B_g}-s_g)$ and $(1-s_{\mathsf{out}})$ are in $\{-1,0,1\}$, we have $\sum_{g\in\mathsf{Gates}}(1-s_{A_g}\cdot s_{B_g}-s_g)^2+(1-s_{\mathsf{out}})^2 < p$, and thus this sum is equal to 0 modulo p if and only if all terms $(1-s_{A_g}\cdot s_{B_g}-s_g)$ and $(1-s_{\mathsf{out}})$ are 0. Therefore Equation (8) is equivalent to the condition that

$$\sum_{g\in\mathsf{Gates}} (1-s_{A_g}\cdot s_{B_g}-s_g)^2+(1-s_{\mathsf{out}})^2 = 0 \mod p.$$

Hence there exists $\{s_g\}_{g\in\mathsf{Gates}} \in \{0,1\}^{|\mathsf{Gates}|}$ such that $P_{C(x,\cdot)}(\{s_g\}_{g\in\mathsf{Gates}}) := 1+\sum_{g\in\mathsf{Gates}}(1-s_{A_g}\cdot s_{B_g}-s_g)^2 + (1-s_{\mathsf{out}})^2 = 1 \mod p$ if and only if there exists $w\in\{0,1\}^m$ such that $C(x,w)=1$. Here, we remark that the degree of $P_{C(x,\cdot)}$ is at most 4, which is in particular a constant.

Next, we remark that if $\mathsf{SKE.Dec}$ can be computed in $\mathbf{NC}^1$, then it can be expressed as a polynomial over $\mathbb{Z}_p$ of degree $D_{\mathsf{Dec}}(\kappa)$ that is polynomial in κ. Based on these observations, if we set $\mathsf{ct}\leftarrow\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}},\{s_g\}_{g\in\mathsf{Gates}})$ instead of $\mathsf{ct}\leftarrow\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}},w)$, and we slightly modify the definition of $f_{x,\mathsf{ct}}$ as $f_{x,\mathsf{ct}}(\mathsf{K}_{\mathsf{SKE}}) := P_{C(x,\cdot)}(\mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}},\mathsf{ct}))$ in the construction given in Appendix C.2, then the degree of $f_{x,\mathsf{ct}}$ is $4D_{\mathsf{Dec}}(\kappa)$, and there exists $\mathsf{K}_{\mathsf{SKE}}$ such that $f_{x,\mathsf{ct}}(\mathsf{K}_{\mathsf{SKE}}) = 1 \mod p$ if and only if there exists $w\in\{0,1\}^m$ such that $C(x,w)=1$. Thus we can instantiate the above construction based on a HomAuth scheme that supports polynomials over $\mathbb{Z}_p$ of degree at most $4D_{\mathsf{Dec}}(\kappa)$.

As such a HomAuth scheme, we use the one constructed in Section 5. The size of an evaluated signature of the scheme is $\mathsf{poly}(\kappa, D)$ where D denotes the degree of the polynomial to evaluate. Here, since the degree of $f_{x,\mathsf{ct}}$ is $2D_{\mathsf{Dec}}(\kappa)$, which is polynomial in κ and independent of the size of $|C|$, the size of the signature contained in the above PP-NIZK scheme is $\mathsf{poly}(\kappa)$ that is independent of $|C|$. Hence the whole size of the proof is $|C|+\mathsf{poly}(\kappa)$ since the size of $\mathsf{ct}\leftarrow\mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}},\{s_g\}_{g\in\mathsf{Gates}})$ is $|C|+\mathsf{poly}(\kappa)$. We note that this construction can naturally be generalized to PP-NIZK for unbounded languages without increasing the proof size. In summary, we obtain Theorem 5.13. In particular, our PP-NIZK is statistical sound and non-programmable CRS zero-knowledge.

C.4 PP-NIZK for Leveled Relations with Sublinear Proof Size

Here, we give a variant of the Kim-Wu construction whose proof size is sublinear in the size of the circuit that computes the **NP** relation to prove from a compact HomAuth scheme. This construction only works for **NP** languages with "leveled" relation, which is a relation that can be expressed by a leveled circuit, i.e., a circuit whose gates are divided into L levels, and all incoming wires to a gate of level $i+1$ come from gates of level i. For this case, the proof size of the scheme becomes $|w|+|C|/\log\kappa+\mathsf{poly}(\kappa)$. To the best of our knowledge, this is the first PP-NIZK with sublinear proof size without using FHE or non-falsifiable assumptions.

Leveled Circuits and Relations. First, we define leveled circuits and its "special" levels following [BGI16]. We say that a circuit is a leveled circuit of depth D if its gates are partitioned into $D+1$ levels, all input gates are of level 0, all output gates are of level $D+1$, and all incoming wires to a gate of level $i+1$ come from gates of level i for each $i\in[D]$. The width at level i is defined to be the number of gates of level i. For a leveled circuit C of depth D, we define a set $\mathcal{S}_C\subset\{0,...,D+1\}$ of "special" levels in the following manner. For each $j\in\{0,...,\lfloor D/\log\kappa\rfloor-1\}$, $\mathcal{S}_C$ contains one level i in the interval $[j\log\kappa+1,...,(j+1)\log\kappa]$ such that the width at level i is the minimum among the width at levels in this interval. (If there exist multiple levels whose width are minimum, we choose the smallest level.) We say that i is a special level if $i\in\mathcal{S}_C$. Let $\mathsf{pre}(i)$ denote the precedent special level of i, i.e., the maximal $i'<i$ such that $i'\in\mathcal{S}_C$ (if such i' does not exist, then we define $\mathsf{pre}(i):=0$) and L_C denote the largest special level of C, i.e., the largest i such that $i\in\mathcal{S}_C$. It is easy to see that the number of gates of a special level is at most $|C|/\log\kappa$ since $\mathcal{S}_C$ contains levels whose width are the smallest in the corresponding interval of length $\log\kappa$. For

any gate g of a special level $i \in \mathcal{S}_C$, we can compute the output value of g as a function of output values of gates of level $\mathsf{pre}(i)$. We denote this function by $\mathsf{EvalfromPre}_g$. Since each special level is at most $2\log\kappa$ far apart from its precedent special level, $\mathsf{EvalfromPre}_g$ can be expressed as a circuit of depth at most $2\log\kappa$. Similarly, we define a function $\mathsf{EvalfromPre}_{\mathsf{out}}$ to be a function that computes the output value of C given output values gates of level L_C as input. Similarly, $\mathsf{EvalfromPre}_{\mathsf{out}}$ can be expressed as a circuit of depth at most $2\log\kappa$.

An **NP** relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is said to be a *leveled relation* if there exists a family $\{C_{n,m} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}\}$ of leveled circuits such that for $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$, we have $C_{n,m}(x,w) = 1$ if and only if $(x,w) \in \mathcal{R}$. In the following, we fix n and m, and omit the subscripts n and m from C for notational simplicity. For $x \in \{0,1\}^n$, we let $\mathsf{SGates}[C(x,\cdot)]$ be the set of all gates of $C(x,\cdot)$ whose level is a special level. For a gate g of $C(x,\cdot)$, we let s_g be the output value of the gate g when $C(x,\cdot)$ is evaluated on input w. We call $w' := (w, \{s_g\}_{g\in\mathsf{SGates}[C(x,\cdot)]})$ an expanded witness of w w.r.t. x and C. It is easy to see that we have $|w'| \le |w| + |C|/\log\kappa$ since $|\mathsf{SGates}[C(x,\cdot)]|$ is at most $|C|/\log\kappa$. Then we define an expanded circuit $\mathsf{ExpCir}_{C(x,\cdot)}$ for the expanded witness as follows.

$\mathsf{ExpCir}_{C(x,\cdot)}(w')$**:** It parses $(w, \{s_g\}_{g\in\mathsf{SGates}[C(x,\cdot)]}) \leftarrow w'$. For all $i \in \mathcal{S}_C$, we denote the output values of gates of level i (in a canonical order) by S_i and we let $S_0 := w$. For all gates g of a special level $i \in \mathcal{S}_C$, it verifies if $s_g = \mathsf{EvalfromPre}_g(S_{\mathsf{pre}(i)})$ holds, and returns 0 if this does not hold. If all check pass, it outputs $\mathsf{EvalfromPre}_{\mathsf{out}}(S_{L_{C(x,\cdot)}})$.

It is easy to see that for any $x \in \{0,1\}^n$, there exists an expanded witness w' such that $\mathsf{ExpCir}_{C(x,\cdot)}(w') = 1$ if and only if there exists a witness $w \in \{0,1\}^m$ such that $C(x,w) = 1$. We can implement $\mathsf{ExpCir}_{C(x,\cdot)}$ by a circuit of depth at most $2\log\kappa + \log(|C|/\log\kappa + 1)$. This can be seen by observing that $\mathsf{ExpCir}_{C(x,\cdot)}$ first computes $\mathsf{EvalfromPre}_g$ for at most $|C|/\log\kappa$ different g and $\mathsf{EvalfromPre}_{\mathsf{out}}$, each of which can be computed by a circuit of depth at most $2\log\kappa$, followed by taking the AND of them. Since the last AND is fan-in at most $|C|/\log\kappa + 1$, this can be implemented by a circuit of depth $\log(|C|/\log\kappa + 1)$ and fan-in 2. Especially, if $|C| = \mathsf{poly}(\kappa)$, then there exists a constant c such that $\mathsf{ExpCir}_{C(x,\cdot)}$ can be computed by a circuit of depth at most $c\log\kappa$.

Then we instantiate the construction given in Appendix C.2 with the following setting.

- Let $n(\kappa)$ and $m(\kappa)$ be any fixed polynomials. Let $\mathcal{L}$ be an **NP** language defined by a leveled relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$. Namely, there exists a leveled circuit C such that for all $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$, we have $C(x,w) = 1$ if and only if $(x,w) \in \mathcal{R}$. We assume that $\mathsf{ExpCir}_{C(x,\cdot)}$ as defined in the previous paragraph can be computed by a circuit of depth at most $c\log\kappa$ for a constant c for all $x \in \{0,1\}^n$.
- Let $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric key encryption scheme with ciphertext space $\mathcal{CT}$ and key space $\{0,1\}^\ell$ whose ciphertext overhead (i.e., ciphertext length minus message length) is $\mathsf{poly}(\kappa)$ where poly is a polynomial independent of the message length, and whose decryption algorithm is computed by a circuit of depth at most $D_{\mathsf{Dec}}(\kappa)$
- For $x \in \{0,1\}^n$ and $\mathsf{ct} \in \mathcal{CT}$, we define a function $f_{x,\mathsf{ct}}(\mathsf{K}_{\mathsf{SKE}}) := \mathsf{ExpCir}_{C(x,\cdot)}(\mathsf{SKE.Dec}(\mathsf{K}_{\mathsf{SKE}}, \mathsf{ct}))$. We note that the depth of $f_{x,\mathsf{ct}}$ is at most $c\log\kappa + D_{\mathsf{Dec}}(\kappa)$ for every $x \in \{0,1\}^n$ and $\mathsf{ct} \in \mathcal{CT}$.
- Let $\mathsf{HA} = (\mathsf{HA.KeyGen}, \mathsf{HA.Sign}, \mathsf{HA.Eval}, \mathsf{HA.VerifyFresh}, \mathsf{HA.VerifyEvaled})$ be a compact HomAuth scheme that supports all circuits of depth at most $c\log\kappa + D_{\mathsf{Dec}}(\kappa)$. Recall that "compact" means that the size of an evaluated signature is $\mathsf{poly}(\kappa)$ that does not depend on the message length or the function to evaluate.
- We set $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w')$ instead of $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{K}_{\mathsf{SKE}}, w)$ where w' is the expanded witness of w w.r.t. x and C.

Since we simply changed the language to prove in the construction given in Appendix C.2, the security can be proven similarly.

Proof Size. In the above construction, a proof consists of an encryption ct of the expanded witness w' and an evaluated signatures of Π_{HA}. The size of expanded witness is at most $|w| + |C|/\log\kappa$, and therefore the size of ct is at most $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$. The size of the signature is at most $\mathsf{poly}(\kappa)$ by the assumed compactness of Π_{HA}. In summary, the proof size of the above scheme is at most $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$.

Instantiations. The above scheme can be instantiated based on any compact HomAuth schemes. They include the following schemes:

1. The HomMAC based on the CDHI assumption proposed by Catalano and Fiore [CF18].
2. The HomSig given in Section 4.2 based on the CDHER assumption. Since this scheme is publicly verifiable, we can make the above scheme publicly verifiable (i.e., DP-NIZK).
3. The HomMAC given in Section 5 based on the DDH assumption. Though the HomMAC is not compact, we can achieve sublinear proof size by using the technique presented in Appendix C.3 as sketched below. First, we expand a witness w to generate an expanded witness $w' = (w, \{s_g\}_{g\in\mathsf{SGates}})$. Then $C(x,w)$ can be computed as
$$\prod_{g\in\mathsf{SGates}} (1-(s_g - \mathsf{EvalfromPre}_g(S_{\mathsf{pre}(i_g)}))^2)\cdot(1-(1-\mathsf{EvalfromPre}_{\mathsf{out}}(S_{L_{C(x,\cdot)}}))^2)$$
where i_g denotes g's level. By using a similar trick to the one used in Appendix C.3, the condition that $C(x,w)=1$ is equivalent to the condition that
$$\begin{aligned} P_{C(x,\cdot)}(w') &:= 1 + \sum_{g\in\mathsf{SGates}} (s_g - \mathsf{EvalfromPre}_g(S_{\mathsf{pre}(i_g)}))^2 + (1-\mathsf{EvalfromPre}_{\mathsf{out}}(S_{L_{C(x,\cdot)}}))^2 \\ &= 1 \mod p. \end{aligned}$$
Since the degrees of $\mathsf{EvalfromPre}_g$ $\mathsf{EvalfromPre}_{\mathsf{out}}$ are at most κ^2 as they are implemented by a circuit of depth at most $2\log\kappa$, the degree of $P_{C(x,\cdot)}$ is $\mathsf{poly}(\kappa)$, which does not depend on $|C|$. Hence, the degree of the function that is evaluated on signatures in the construction of PP-NIZK is $\mathsf{poly}(\kappa)$, and thus the size of evaluated signature is $\mathsf{poly}(\kappa)$. (Recall that the evaluated signature size in HomMAC given in Section 5 only depends on the degree of the function, and does not depend on the size of the function.) Then the whole proof size is $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$.
4. The HomSig scheme based on the LWE assumption proposed by Gorubnov, Vaikuntanathan, and Wichs [GVW15b] (or its variant by Kim and Wu [KW18a]). We note that since the HomSig of [GVW15b] supports all polynomial-size circuits and have (weak)-compactness, the proof sizes of the original Kim-Wu construction with the HomSig are $|w| + \mathsf{poly}(\kappa, d)$ where d is the depth of the circuit computing the **NP** relation. Therefore, unless d is very large, the conversion presented in this subsection may not be useful for the LWE-based HomSig scheme.

C.5 Bounded to Unbounded PP-NIZK

Here, we briefly remark that we can convert any PP-NIZK for bounded languages to the one for unbounded languages. Let $\mathcal{L}$ be an **NP** language defined by a relation $\mathcal{R}$, which can be computed by a family of circuits $\{C_{n,m}\}_{n,m\in\mathbb{N}}$. Namely, for all $x\in\{0,1\}^n$ and $w\in\{0,1\}^m$, $C_{n,m}(x,w)=1$ if and only if $(x,w)\in\mathcal{R}$. We explain how to generate a proof for $\mathcal{R}$ without a priori fixing n and m. Let Π_{PPNIZK} be a PP-NIZK for bounded languages, and we construct Π'_{PPNIZK}, which is a PP-NIZK for unbounded languages. To make this conversion, we additionally use a commitment scheme. The main idea is as follows: The proving algorithm Π'_{PPNIZK} given $x\in\{0,1\}^n$ and $w\in\{0,1\}^m$, computes an expanded witness $\{s_g\}_{g\in\mathsf{Gates}}$ where Gates is the set of all gates of $C_{n,m}(x,\cdot)$, and s_g is the output value of the gate g when $C_{n,m}(x,\cdot)$ is evaluated on the input w. Then it generates a commitment c of the expanded witness $\{s_g\}_{g\in\mathsf{Gates}}$ in a bit-by-bit manner. Then it generates a proof of the consistency of $\{s_g\}_{g\in\mathsf{Gates}}$ for each gate of $C_{n,m}(x,\cdot)$ by using the proving algorithm of Π_{PPNIZK}. Here, each execution of the proving algorithm of Π_{PPNIZK} simply verifies that at most 3 committed values are consistent with the corresponding gate of $C(x,\cdot)$. Especially, it is clear that the complexity of this consistency check is independent of n and m. It is easy to reduce the soundness and the zero-knowledge property of Π'_{PPNIZK} to those of Π_{PPNIZK} and the security of the underlying commitment scheme.

D Homomorphic Authenticators with Online-Offline / Amortized Verification Efficiency

In this section, we show that our HomSig construction for $\mathbf{NC}^1$ in Section 4.2 can be extended to the multi-data setting. Furthermore, the construction provides online-offline and amortized verification efficiency. Notably, in the former, the offline phase of the verification algorithm only consists of a constant number of pairing operations.

D.1 Multi-Data Homomorphic Authenticators

In this section, we provide the definition of a *multi-data* homomorphic authentication scheme with *online-offline efficiency* [BFR13, GVW15b] and show that the HomAuth scheme in Section 4 satisfies this notion with minor modifications. In a multi-data HomAuth scheme, each set of data is assigned a unique data set identifier Δ. The simple HomAuth scheme we defined in Section 2.3 is referred to as a single-data HomAuth. The advantage of a multi-data HomAuth over a single-data HomAuth is that the verifier only needs to perform work proportional to the circuit size of C once to verify arbitrarily many signatures evaluated on C under different data set identifiers and signed by any signers. Specifically, an online-offline efficient multi-data offers a much better amortized efficiency for the verifier.

Let $\{\mathcal{X}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of message spaces. Let $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of circuits, where $\mathcal{C}_\kappa$ is a set of polynomial sized circuits with domain $\mathcal{X}_\kappa^{\ell(\kappa)}$ and range $\mathcal{X}_\kappa$ whose depth is bounded by $d(\kappa)$. In the following, we occasionally drop the subscript when the meaning is clear. In case $\mathcal{X} = \{0,1\}$, we set $\mathcal{C}$ to be simply a family of boolean circuits.

Below, we define a single-shot multi-data homomorphic authenticator scheme. Let $\mathcal{L} \subseteq \{0,1\}^*$ be the set of data set identifiers. Let $\{\Sigma_{\mathsf{Fresh}_\kappa}\}_{\kappa\in\mathbb{N}}$ and $\{\Sigma_{\mathsf{Evaled}_\kappa}\}_{\kappa\in\mathbb{N}}$ be families of signature spaces, where each of them corresponds to the output space of fresh signatures and evaluated signatures, respectively.

Definition D.1 (Multi-Data Homomorphic Authenticators). *A multi-data homomorphic authenticator (HomAuth) scheme Π_{HA} with message space $\mathcal{X}$ for the circuit class $\mathcal{C}$ is defined by the following algorithms:*

$\mathsf{HA.KeyGen}(1^\kappa, 1^\ell) \to (\mathsf{vk}, \mathsf{ek}, \mathsf{sk})$*: The key generation algorithm takes as input the security parameter 1^κ and the message length 1^ℓ and outputs a verification key* vk*, an evaluation key* ek*, and a signing key* sk*.*

$\mathsf{HA.Sign}(\mathsf{sk}, \Delta, \mathbf{x} = (x_1, \cdots, x_\ell)) \to \boldsymbol{\sigma}$*: The signing algorithm takes as input a signing key* sk*, a data set identifier $\Delta \in \mathcal{L}$, and messages $\mathbf{x} \in \mathcal{X}^\ell$, and outputs a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$.*

$\mathsf{HA.Eval}(\mathsf{ek}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma}) \to \sigma$*: The signature-evaluation algorithm takes as input an evaluation key* ek*, a data set identifier $\Delta \in \mathcal{L}$, a circuit $C : \mathcal{X}^\ell \to \mathcal{X}$ in $\mathcal{C}$, messages $\mathbf{x} \in \mathcal{X}^\ell$, and a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$ and outputs an evaluated signature $\sigma \in \Sigma_{\mathsf{Evaled}}$.*

$\mathsf{HA.VerifyFresh}(\mathsf{vk}, \Delta, \mathbf{x}, \boldsymbol{\sigma}) \to \top$ *or* $\bot$*: The fresh verification algorithm takes as input a verification key* vk*, a data set identifier $\Delta \in \mathcal{L}$, messages $\mathbf{x} \in \mathcal{X}^\ell$, and a signature $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$, and outputs $\top$ if the signature is valid and outputs $\bot$ otherwise.*

$\mathsf{HA.VerifyEvaled}(\mathsf{vk}, \Delta, C, z, \sigma) \to \top$ *or* $\bot$*: The evaluated verification algorithm splits into a pair of algorithms (*$\mathsf{HA.VerifyEvaledOffL}$, $\mathsf{HA.VerifyEvaledOnL}$*):*

- $\mathsf{HA.VerifyEvaledOffL}(\mathsf{vk}, C) \to \mathsf{vk}_C$*: The offline evaluated verification algorithm takes as input a verification key* vk *and a circuit $C : \mathcal{X}^\ell \to \mathcal{X}$ in $\mathcal{C}$, and outputs an evaluated verification key* vk_C*.*
- $\mathsf{HA.VerifyEvaledOnL}(\mathsf{vk}', \Delta, z, \sigma) \to \top$ *or* $\bot$*: The online evaluated verification algorithm takes as input an (evaluated) verification key* vk'*, a data set identifier $\Delta \in \mathcal{L}$, a message $z \in \mathcal{X}$, and a signature $\sigma \in \Sigma_{\mathsf{Evaled}}$, and outputs $\top$ if the signature is valid and outputs $\bot$ otherwise.*

Correctness. We say a multi-data homomorphic authentication scheme Π_{HA} is correct, if for all $\kappa \in \mathbb{N}$, $\ell \in \mathsf{poly}(\kappa)$, data set identifiers $\Delta \in \mathcal{L}$, messages $\mathbf{x} = (x_1, \cdots, x_\ell) \in \mathcal{X}^\ell$, and $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \in \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$ the following two conditions hold:

(1) Signing Correctness: For all $\boldsymbol{\sigma} \in \mathsf{HA.Sign}(\mathsf{sk}, \Delta, \mathbf{x})$ in Σ_{Fresh}, we have

$$\Pr[\mathsf{HA.VerifyFresh}(\mathsf{vk}, \Delta, \mathbf{x}, \boldsymbol{\sigma}) = \top] = 1.$$

(2) Evaluation Correctness: For all circuits $C \in \mathcal{C}$, signatures $\boldsymbol{\sigma}$ such that $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \Delta, \mathbf{x}, \boldsymbol{\sigma}) = \top$, $\sigma \in \mathsf{HA.Eval}(\mathsf{ek}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma})$ in Σ_{Evaled}, and $\mathsf{vk}_C \in \mathsf{HA.VerifyEvaledOffL}(\mathsf{vk}, C)$ we have

$$\Pr[\mathsf{HA.VerifyEvaledOnL}(\mathsf{vk}_C, \Delta, C(\mathbf{x}), \sigma) = \top] = 1.$$

Compactness and Online-offline Efficiency. We say a multi-data HomAuth scheme Π_{HA} is *compact* if there exists a universal polynomial $\mathsf{poly}(\cdot)$ such that for all valid fresh data set identifier-message-signature pair $(\Delta, \mathbf{x}, \boldsymbol{\sigma}) \in \mathcal{L} \times \mathcal{X}^{\ell} \times \Sigma_{\mathsf{Fresh}}$ and circuit $C \in \mathcal{C}$, the size of the evaluated signature $\sigma_C \leftarrow \mathsf{HA.Eval}(\mathsf{ek}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma})$ satisfies $|\sigma_C| \leq \mathsf{poly}(\kappa)$. We say Π_{HA} is *weakly-compact* if $|\sigma_C| \leq \mathsf{poly}(\kappa, d)$ instead, where d is the maximum depth of the circuit in $\mathcal{C}$.

Furthermore, we say Π_{HA} is *online-offline efficient* if there exists a universal polynomial $\mathsf{poly}(\cdot)$ such that the running time of $\mathsf{HA.VerifyEvaledOnL}(\mathsf{vk}_C, \Delta, C(\mathbf{x}), \sigma_C)$ is $\mathsf{poly}(\kappa)$ for any data set identifier $\Delta \in \mathcal{L}$, messages $\mathbf{x} \in \mathcal{X}^{\ell}$, circuit $C \in \mathcal{C}$, signatures $\boldsymbol{\sigma}$ such that $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \Delta, \mathbf{x}, \boldsymbol{\sigma}) = \top$, $\sigma_C \in \mathsf{HA.Eval}(\mathsf{ek}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma})$ in Σ_{Evaled}, and evaluated verification key $\mathsf{vk}_C \in \mathsf{HA.VerifyEvaledOffL}(\mathsf{vk}, C)$. Notably, regardless of the associated data set identifier, the time it takes to verify an evaluated signature on circuit C should be much smaller than the time it takes to compute C once the verification key has been preprocessed. Here, we allow the running time of the offline phase, i.e., the time it takes to run $\mathsf{vk}_C \leftarrow \mathsf{HA.VerifyEvaledOffL}(\mathsf{vk}, C)$, to depend on the time it takes to compute C. Above, in case the running time is bounded by $\mathsf{poly}(\kappa, d)$, where d is the maximum depth of the circuit in $\mathcal{C}$, we say it is *weakly* online-offline efficient.

(Single-Shot) Unforgeability. We now define single-shot multi-data unforgeability for a HomAuth scheme, where the adversary must declare the challenge messages all at once. Below we assume that checking membership of $\mathcal{C}, \Sigma_{\mathsf{Fresh}}, \Sigma_{\mathsf{Evaled}}$ can be done efficiently. The security notion is defined formally by the following game between a challenger and an adversary $\mathcal{A}$.

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given 1^{κ} as input and sends 1^{ℓ} to the challenger. Then the challenger generates a key pair $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{HA.KeyGen}(1^{\kappa}, 1^{\ell})$ and gives ek to $\mathcal{A}$. In case it is an HomSig scheme, i.e., the signatures are publicly verifiable, then the challenger further provides vk to $\mathcal{A}$. Finally, the challenger prepares an empty list T.

Signing Query: The adversary $\mathcal{A}$ can ask an arbitrary number of signing queries. In each query, $\mathcal{A}$ submits a set of messages $\mathbf{x} \in \mathcal{X}^{\ell}$ and a data set identifier $\Delta \in \mathcal{L}$ to be signed. If $\Delta \in T$, the challenger aborts. Otherwise, the challenger responds by creating a signature $\boldsymbol{\sigma} \leftarrow \mathsf{HA.Sign}(\mathsf{sk}, \Delta, \mathbf{x})$ and sends $\boldsymbol{\sigma} \in \Sigma_{\mathsf{Fresh}}$ to $\mathcal{A}$. Finally, the challenger adds Δ to T.

Verification Query: The adversary $\mathcal{A}$ may adaptively query a data set identifier-message-signature(-circuit) pair. When the query is of type $(\Delta, \mathbf{x}, \boldsymbol{\sigma}) \in \mathcal{L} \times \mathcal{X}^{\ell} \times \Sigma_{\mathsf{Fresh}}$, the challenger returns the output of $\mathsf{HA.VerifyFresh}(\mathsf{vk}, \Delta, \mathbf{x}, \boldsymbol{\sigma})$. When the query is of type $(\Delta, z, \sigma, C) \in \mathcal{L} \times \mathcal{X} \times \Sigma_{\mathsf{Evaled}} \times \mathcal{C}$, the challenger returns the output of $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, \Delta, C, z, \sigma)$. For any other types of queries, the challenger returns $\perp$.[11]

Forgery: Then the adversary $\mathcal{A}$ outputs a circuit $C^{\star}$, a data set identifier $\Delta^{\star}$, a message $z^{\star} \in \mathcal{X}$, and a signature $\sigma^{\star}$ as the forgery. We say that $\mathcal{A}$ wins the game if $C^{\star} \in \mathcal{C}$, $\sigma^{\star} \in \Sigma_{\mathsf{Evaled}}$, $\mathsf{HA.VerifyEvaled}(\mathsf{vk}, \Delta^{\star}, C^{\star}, z^{\star}, \sigma^{\star}) = \top$ and either:

- Type 1 forgery: $\Delta^{\star} \notin T$, i.e., no signing query with label $\Delta^{\star}$ was ever made.
- Type 2 forgery: $\Delta^{\star} \in T$ and $C^{\star}(\mathbf{x}) \neq z^{\star}$, where $\mathbf{x}$ is the associated message of the data set identifier $\Delta^{\star}$ which was submitted by the adversary during the signing query, i.e., a signing query with label $(\Delta^{\star}, \mathbf{x})$ was made, but the verified signature verifies on a false evaluation output.

The advantage of an adversary winning the above game is defined by $\Pr[\mathcal{A} \text{ wins}]$, where the probability is taken over the randomness used by the challenger and the adversary. A multi-data homomorphic authenticator scheme Π_{HA} is said to satisfy (single-shot) *statistical* unforgeability if the advantage of any (possibly inefficient) adversary $\mathcal{A}$ is negligible. In case it only holds for adversaries that are computationally bounded, we say it satisfies *computational* unforgeability.

Context-Hiding. We now define context-hiding for a multi-data HomAuth scheme. Since our proposed scheme satisfies the stronger statistical notion of context-hiding, we only provide a simulation-based notion for succinctness of presentation. The computational variant can be defined similarly to the single-data definition in Section 2.3.

[11] Note that if we consider an HomSig scheme, this item can be dismissed since the verification step can be executed by the adversary himself.

A multi-data homomorphic authenticator scheme Π_{HA} is statistically context-hiding if for all $\kappa \in \mathbb{N}$, $\ell \in \mathsf{poly}(\kappa)$, there exists a PPT simulator $\mathsf{HA.Sim}$ such that, for any $(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}) \in \mathsf{HA.KeyGen}(1^\kappa, 1^\ell)$, $C \in \mathcal{C}$, $\Delta \in \mathcal{L}$, and any pair $(\mathbf{x}, z) \in \{(\mathbf{x}, z) \in \mathcal{X}^\ell \times \mathcal{X} \mid C(\mathbf{x}) = z\}$, $\boldsymbol{\sigma} \in \mathsf{HA.Sign}(\mathsf{sk}, \Delta, \mathbf{x})$, we have

$$\{\sigma \leftarrow \mathsf{HA.Eval}(\mathsf{ek}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma})\} \overset{\text{stat}}{\approx} \{\sigma \leftarrow \mathsf{HA.Sim}(\mathsf{vk}, \mathsf{ek}, \mathsf{sk}, \Delta, C, z)\},$$

where the probability is only over the randomness used by the $\mathsf{HA.Eval}$ and $\mathsf{HA.Sim}$ algorithm. If the above distributions are exactly the same, we say that the scheme is *perfectly* context-hiding.

D.2 Construction of Homomorphic Signatures with Online-Offline Efficiency

In this section, we provide a construction of multi-data FHS for $\mathbf{NC}^1$ circuits from the CDHER assumption. The circuit class dealt with by our scheme here is exactly the same as that in Section 4.2, i.e., $\mathcal{C}^{\mathbf{NC}^1} = \{\mathcal{C}^{\mathbf{NC}^1}_{\kappa,d,\ell,s}\}_{\kappa \in \mathbb{N}}$.

Preparation. We first prepare the definition of pseudorandom functions (PRFs) and digital signature schemes, which we use in our construction of multi-data HomSig.

Definition D.2 (Pseudorandom Function). *A pseudorandom function is defined by a PPT algorithm* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$*, where* $\mathcal{K}$*,* $\mathcal{X}$*, and* $\mathcal{Y}$ *are sets (implicitly) parameterized by the security parameter* κ*, and we further assume* $\mathcal{K}$ *is an efficiently sampleable set. We say* PRF *is* pseudorandom *if the following holds for all PPT adversary* $\mathcal{A}$*:*

$$\left|\Pr[\mathcal{A}^{\mathsf{PRF}(\mathsf{K},\cdot)}(1^\kappa) \to 1] \;-\; \Pr[\mathcal{A}^{\mathcal{O}(\cdot)}(1^\kappa) \to 1]\right|,$$

where $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$ *is a random function that returns uniformly random elements over* $\mathcal{Y}$*.*

Definition D.3 (Digital Signatures). *A digital signature scheme with message space* $\{0,1\}^\ell$ *is a triple of polynomial time algorithms* $\mathsf{DS} = (\mathsf{DS.KeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Vrfy})$ *of the following form:*

- $\mathsf{DS.KeyGen}(1^\kappa) \to (\mathsf{DS.vk}, \mathsf{DS.sk})$: *The key generation algorithm takes as input the security parameter* 1^κ *and outputs a verification key* $\mathsf{DS.vk}$ *and signing key* $\mathsf{DS.sk}$*.*
- $\mathsf{DS.Sign}(\mathsf{DS.sk}, x) \to \sigma_{\mathsf{DS}}$: *The signing algorithm takes as inputs the signing key* $\mathsf{DS.sk}$ *and message* $x \in \{0,1\}^\ell$*, and outputs a signature* σ_{DS}*.*
- $\mathsf{DS.Vrfy}(\mathsf{DS.vk}, x, \sigma_{\mathsf{DS}}) \to \top$ *or* $\bot$: *The verification algorithm takes as inputs the verification key* $\mathsf{DS.vk}$*, message* $x \in \{0,1\}^\ell$ *and signature* σ_{DS}*, and outputs* $\top$ *if the signature is valid and outputs* $\bot$ *otherwise.*

Correctness. We say a digital signature scheme is correct if for all $\kappa, \ell \in \mathsf{poly}(\kappa)$, messages $x \in \{0,1\}^\ell$, $(\mathsf{DS.vk}, \mathsf{DS.sk}) \in \mathsf{DS.KeyGen}(1^\kappa)$, $\sigma_{\mathsf{DS}} \in \mathsf{DS.Sign}(\mathsf{DS.sk}, x)$, we have $\mathsf{DS.Vrfy}(\mathsf{DS.vk}, x, \sigma_{\mathsf{DS}}) = \top$.

Eu-cma Security. The security notion of existential unforgeability under an adaptive chosen message attack (eu-cma) is defined by the following game between an adversary $\mathcal{A}$ and a challenger.

Setup: The challenger runs $(\mathsf{DS.vk}, \mathsf{DS.sk}) \leftarrow \mathsf{DS.KeyGen}(1^\kappa)$ and provides $\mathcal{A}$ the verification key $\mathsf{DS.vk}$.

Signature Queries: When $\mathcal{A}$ submits a message $x \in \{0,1\}^\ell$, the challenger responds by returning $\sigma_{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{DS.sk}, x)$.

Output: Finally, $\mathcal{A}$ outputs a pair $(x^\star, \sigma^\star_{\mathsf{DS}})$. The adversary $\mathcal{A}$ wins if $\mathsf{DS.Vrfy}(\mathsf{DS.vk}, x^\star, \sigma^\star_{\mathsf{DS}}) = \top$ and x^* was not submitted by $\mathcal{A}$ as a signature query.

We define the advantage of an adversary $\mathcal{A}$ as the probability that $\mathcal{A}$ wins the above game, where the probability is taken over the randomness used by the challenger and the adversary. A digital signature scheme is called eu-cma secure if the advantage of the above game is negligible for all PPT adversaries.

Construction of Multi-Data HomSig. Let $\mathsf{PRF}(\cdot,\cdot) : \mathcal{K} \times \mathcal{L} \to \mathbb{Z}_p$ be a pseudorandom function, where $\mathcal{K}$ is the key space. In addition, let $\mathsf{DS} = (\mathsf{DS.KeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Vrfy})$ be a signature scheme and $\ell_{\mathsf{DS}} := \ell_{\mathsf{DS}}(\kappa)$ be the bit length of signatures in the scheme. In the following each signature spaces Σ_{Fresh} and Σ_{Evaled} are set as $\mathbb{G} \times \mathbb{Z}_p \times \{0,1\}^{\ell_{\mathsf{DS}}}$ and $\mathbb{G}^4 \times \{0,1\}^{\ell_{\mathsf{DS}}} \times \{0,1\}$, respectively. We assume that $\mathsf{DS.Sign}$ is deterministic. This can be assumed without loss of generality, since the signing algorithm of any signature scheme can be derandomized by a PRF. Then our construction of online/offline efficient multi-data HomSig is provided as follows:

$\mathsf{HS.KeyGen}(1^\kappa, 1^\ell)$: On input the security parameter 1^κ and the message length 1^ℓ, sample the group description $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, g, e(\cdot,\cdot)) \leftarrow \mathsf{BGGen}(1^\kappa)$. Then, sample $\mathsf{K} \xleftarrow{\$} \mathcal{K}$, $(\mathsf{DS.vk}, \mathsf{DS.sk}) \xleftarrow{\$} \mathsf{DS.KeyGen}(1^\kappa)$, $a \leftarrow \mathbb{Z}_p$, and $b_i, c_i \leftarrow \mathbb{Z}_p^*$ for $i \in [n]$ and output

$$\mathsf{HS.vk} = \begin{pmatrix} \left\{g^{a^j}\right\}_{j\in[m]}, & \{g^{c_i}\}_{i\in[n]}, & \left\{g^{a^j/b_i}\right\}_{i\in[n],j\in[2m]\atop j\neq m+1}, & \left\{g^{a^{m+1}c_{i'}/b_ic_i}\right\}_{i,i'\in[n],i\neq i'}, \\ \mathsf{DS.vk}, & \{g^{ac_i}\}_{i\in[n]}, & \left\{g^{a^j/b_ic_i}\right\}_{i\in[n],j\in[2m+1]}, & \left\{g^{a^jc_{i'}/b_i}\right\}_{i,i'\in[n],j\in[m]} \end{pmatrix} \tag{9}$$

and

$$\mathsf{HS.sk} = \left(\mathsf{K}, \mathsf{DS.sk}, a, \{b_i, c_i\}_{i\in[n]}\right).$$

For notational simplicity, we assume the group description $\mathcal{G}$ is implicitly included both in $\mathsf{HS.vk}$ and $\mathsf{HS.sk}$.

$\mathsf{HS.Sign}(\mathsf{HS.sk}, \Delta, \mathbf{x} = (x_1, \cdots, x_\ell))$: On input $\Delta \in \mathcal{L}$ and $x \in \{0,1\}^\ell$, first compute $u := \mathsf{PRF}(\mathsf{K}, \Delta)$ and g^u. Then, run $\sigma_{\mathsf{DS}} \xleftarrow{\$} \mathsf{DS.Sign}(\mathsf{DS.sk}, g^u \| \Delta)$ and $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$ and compute $\tilde{u} \in \mathbb{Z}_p$ as

$$\tilde{u} = u - \sum_{i\in[n]} y_i \cdot \left(a^{m+1}/b_ic_i\right), \tag{10}$$

where $y_i \in \{0,1\}$ is the i-th bit of $\mathbf{y}$. Finally, output $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}}) \in \Sigma_{\mathsf{Fresh}}$.

$\mathsf{HS.Eval}(\mathsf{HS.vk}, \Delta, C, \mathbf{x}, \boldsymbol{\sigma})$: If $\mathbf{x} \notin \{0,1\}^\ell$, $\Delta \notin \mathcal{L}$, $C \notin \mathcal{C}^{\mathbf{NC}^1}$ or $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}}) \notin \Sigma_{\mathsf{Fresh}}$, abort. Otherwise, compute $z = C(\mathbf{x}) \in \{0,1\}$ and construct the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ defined as in Section 4.2, i.e., $\tilde{C}_z(x)$ outputs the bool-value of $(C(\mathbf{x}) \stackrel{?}{=} z)$. Here, we have $\tilde{C}_z(\mathbf{x}) = 1$ by construction. Then, run $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$ and $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M} \in \mathbb{Z}_p^{n\times m}$. By Lemma 4.5, $\mathbf{y}$ *does not* satisfy the span program $\mathbf{M}$ since $\tilde{C}_z(\mathbf{x}) = 1$. Then find a vector $\mathbf{d} = (d_1, \ldots, d_m)^\top \in \mathbb{Z}_p^m$ such that $d_1 = -1$ and $\langle \mathbf{M}_i, \mathbf{d}\rangle = 0$ for all $i \in [n]$ satisfying $y_i = 0$, where $\mathbf{M}_i$ is the i-th row of $\mathbf{M}$. Note that such a vector exists and can be found efficiently due to Lemma 4.4. Then pick $\tilde{r} \xleftarrow{\$} \mathbb{Z}_p$ and compute K_1, K_2, and K_3 as Eq. (2), (3), and (4) in Section 4.2. Finally, output $\sigma = (K_1, K_2, K_3, g^u, \sigma_{\mathsf{DS}}, z) \in \Sigma_{\mathsf{Evaled}}$.

$\mathsf{HS.VerifyFresh}(\mathsf{HS.vk}, \mathbf{x}, \Delta, \boldsymbol{\sigma})$: Output $\perp$ if $\mathbf{x} \notin \{0,1\}^\ell$ or $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}}) \notin \Sigma_{\mathsf{Fresh}}$. Otherwise, first run $\mathsf{EncInp}(\mathbf{x}) = \mathbf{y} \in \{0,1\}^n$. Then, check the following condition:

$$g^u \stackrel{?}{=} g^{\tilde{u}} \cdot \prod_{i\in[n]} \left(g^{a^{m+1}/b_ic_i}\right)^{y_i}, \qquad \mathsf{DS.Vrfy}(\mathsf{DS.vk}, g^u\|\Delta, \sigma_{\mathsf{DS}}) \stackrel{?}{=} \top,$$

where $y_i \in \{0,1\}$ is the i-th bit of $\mathbf{y}$. If they hold output $\top$, otherwise output $\perp$.

$\mathsf{HS.VerifyEvaledOffL}(\mathsf{HS.vk}, C)$: Output $\perp$ if $C \notin \mathcal{C}^{\mathbf{NC}^1}$. Otherwise, construct the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ and run $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M}_z \in \mathbb{Z}_p^{n\times m}$ for $z \in \{0,1\}$. Then compute $e(g,g)^{a^{m+1}} = e(g^a, g^{a^m})$ and

$$V_z := \prod_{i\in[n],j\in[m]} \left(g^{a^j/b_i}\right)^{M_{z,i,j}}$$

for $z \in \{0,1\}$, where $M_{z,i,j}$ is the (i,j)-th entry of $\mathbf{M}_z$. Finally output $\mathsf{HS.vk}_C = (g^a, e(g,g)^{a^{m+1}}, V_0, V_1)$.

$\mathsf{HS.VerifyEvaledOnL}(\mathsf{HS.vk}_C, \Delta, z, \sigma)$: Parse $\mathsf{HS.vk}_C = (e(g,g)^{a^{m+1}}, V_0, V_1)$ and $\sigma = (K_1, K_2, K_3, g^u, \sigma_{\mathsf{DS}}, z')$ and output $\perp$ if any of the following holds: $z \notin \{0,1\}$, $z \neq z'$, and $\sigma \notin \Sigma_{\mathsf{Evaled}}$. Then, check the following conditions:

$$e\left(K_1,\ g^u \cdot V_z\right) \stackrel{?}{=} e(K_3, g), \qquad e(g, K_2) \cdot e(g^a, K_1)^{-1} \stackrel{?}{=} e(g,g)^{a^{m+1}},$$
$$\mathsf{DS.Vrfy}(\mathsf{DS.vk}, g^u\|\Delta, \sigma_{\mathsf{DS}}) \stackrel{?}{=} \top.$$

If the above equations hold, output $\top$. Otherwise output $\perp$.

Correctness. The correctness follows from that of DS and by a similar argument to the case of single-data scheme.
Online-Offline Efficiency. We can see that the online phase of the verification algorithm roughly consists of only 4 pairing computations plus the execution of DS.Vrfy, which is independent from the size of the circuit. For example, if we instantiate DS with the Waters' signature scheme [Wat05], the total cost of the verification in the online phase will be about 6 pairing computations.

D.3 Security Proof

Here, we prove that our construction is context-hiding and single-shot unforgeable.

Theorem D.4. *Our construction is perfectly context-hiding.*

Proof. To show the context-hiding property, we first construct the homomorphic signature simulator HS.Sim as follows:

$\mathsf{HS.Sim}(\mathsf{HS.vk}, \mathsf{HS.sk}, \Delta, C, z)$: On input the signing key HS.sk, a data identifier Δ, a circuit $C \in \mathcal{C}^{\mathbf{NC}^1}$, and a message $z \in \{0,1\}$, it first constructs the circuit $\tilde{C}_z \in \tilde{\mathcal{C}}^{\mathbf{NC}^1}$ associated to circuit C. Then, it computes $\mathsf{EncCir}(\tilde{C}_z) = \mathbf{M} \in \mathbb{Z}_p^{n \times m}$. It also computes $u = \mathsf{PRF}(\mathsf{K}, \Delta)$ and $\sigma_{\mathsf{DS}} = \mathsf{DS.Sign}(\mathsf{DS.sk}, g^u \| \Delta)$. It then picks $r \xleftarrow{\$} \mathbb{Z}_p$ and computes and (K_1, K_2, K_3) as Eq. (5) in Theorem 4.6. Finally, it outputs $\sigma = (K_1, K_2, K_3, g^u, \sigma_{\mathsf{DS}}, z) \in \Sigma_{\mathsf{Evaled}}$.

We now proceed to show that this simulator HS.Sim satisfies the required conditions. Namely, the distribution of σ output from $\mathsf{HS.Sim}(\mathsf{HS.vk}, \mathsf{HS.sk}, \Delta, C, z)$ and that output from $\mathsf{HS.Eval}(\mathsf{HS.vk}, C, \mathbf{x}, \boldsymbol{\sigma})$ for $\boldsymbol{\sigma} \in \mathsf{HS.Sign}(\mathsf{HS.sk}, \Delta, \mathbf{x})$ are exactly the same. We first observe that u is uniquely defined from HS.sk and Δ. This then implies that σ_{DS} is uniquely defined from HS.sk and Δ, where we use our assumption that the signing algorithm DS.Sign is deterministic. Then by exactly the same proof as Theorem 4.6, it can also be seen that the distribution of (K_1, K_2, K_3) computed as Eq. (5) and that computed as Eq. (2), (3), and (4) are exactly the same. This concludes the proof of the theorem. □

Theorem D.5. *Our construction satisfies single-shot unforgeability assuming that* DS *is* eu-cma *secure,* PRF *is pseudorandom, and the* (n, m)*-CDHER assumption holds.*

Proof. We prove the single-shot unforgeability of our multi-data HomSig scheme in Appendix D.2 assuming single-shot partial adaptive unforgeability (See Remark 4.8 for the definition) of our single-data FHS scheme in Section 4.2 and unforgeability of DS. We consider two types of forgery, namely, Type A and Type B. The Type A forgery is of the form $(C^\star, \Delta^\star, z^\star, \sigma^\star = (K_1^\star, K_2^\star, K_3^\star, g^{u^\star}, \sigma_{\mathsf{DS}}^\star, z^\star))$ where $\Delta^\star \notin T$ or $\Delta^\star \in T \wedge u^\star \neq \mathsf{PRF}(K, \Delta^\star)$. On the other hand, a Type B forgery satisfies $\Delta^\star \in T \wedge u^\star = \mathsf{PRF}(K, \Delta^\star)$ and $C^\star(\mathbf{x}) \neq z^\star$, where $\mathbf{x}$ is the corresponding message to the signing query for $\Delta^\star$. Let E_{A} and E_{B} be the event that an adversary $\mathcal{A}$ succeeds in generating Type A and Type B forgeries, respectively. Since Type A and B forgeries cover all cases of the forgery, it suffices to show $\Pr[\mathsf{E}_{\mathrm{A}}] \leq \mathsf{negl}(\kappa)$ and $\Pr[\mathsf{E}_{\mathrm{B}}] \leq \mathsf{negl}(\kappa)$ for any PPT adversary $\mathcal{A}$.

Lemma D.6. *We have* $\Pr[\mathsf{E}_{\mathrm{A}}] = \mathsf{negl}(\kappa)$ *assuming* DS *is* eu-cma *secure.*

Proof. To prove the lemma, it suffices to show that there exists a PPT algorithm $\mathcal{B}$ that breaks eu-cma security of DS with probability $\Pr[\mathsf{E}_{\mathrm{A}}]$. The description of $\mathcal{B}$ is as follows.

At the beginning of the game, $\mathcal{B}$ is given DS.vk from its challenger. Then, $\mathcal{B}$ chooses $\mathsf{K} \xleftarrow{\$} \mathcal{K}$, $a \leftarrow \mathbb{Z}_p$, and $b_i, c_i \leftarrow \mathbb{Z}_p^*$ for $i \in [n]$ and sets the verification key HS.vk as Eq. (9). Then, $\mathcal{B}$ gives HS.vk to $\mathcal{A}$. During the game, $\mathcal{A}$ makes signing queries. When $\mathcal{A}$ makes a query for $(\mathbf{x}, \Delta)$, $\mathcal{B}$ returns $\perp$ if $\Delta \in T$. Otherwise, it computes $u = \mathsf{PRF}(\mathsf{K}, \Delta)$ and makes a signing query for $g^u \| \Delta$ to its challenger. Given σ_{DS} from the challenger, $\mathcal{B}$ computes $\tilde{u}$ as Eq. (10) and returns $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}})$ to $\mathcal{A}$. At the end of the game, the adversary $\mathcal{A}$ outputs a forgery $(C^\star, \Delta^\star, z^\star, \sigma^\star)$. Then, $\mathcal{B}$ checks $C^\star \in \mathcal{C}$, $\sigma^\star \in \Sigma_{\mathsf{Evaled}}$, $\mathsf{HS.VerifyEvaled}(\mathsf{vk}, \Delta^\star, C^\star, z^\star, and \sigma^\star) = \top$, and aborts if they do not hold. It further checks whether $\Delta^\star \notin T$ or $u^\star \neq \mathsf{PRF}(K, \Delta^\star)$, and aborts if they do not hold. If both conditions hold, $\mathcal{B}$ parses $\sigma^\star \to (K_1^\star, K_2^\star, K_3^\star, g^{u^\star}, \sigma_{\mathsf{DS}}^\star, z^\star)$ and outputs $(g^{u^\star} \| \Delta^\star, \sigma_{\mathsf{DS}}^\star)$ as its forgery.

It is easy to observe that $\mathcal{B}$ wins the game if and only if $\mathcal{A}$ generates a Type A forgery. Furthermore, $\mathcal{B}$'s simulation is perfect. This concludes the proof of the lemma. □

Lemma D.7. *We have* $\Pr[\mathsf{E}_{\mathrm{B}}] = \mathsf{negl}(\kappa)$ *assuming our single-data HomSig scheme in Section 4.2 is partial adaptive unforgeable and* PRF *is pseudorandom.*

Proof. We show the lemma by considering the following sequence of games. In the following, let E'_i denote the event that the challenger outputs 1 in Game_i.

Game_0**:** This game is the single-shot unforgeability game for multi-data HomSig. At the end of the game, the challenger outputs 1 if and only if $\mathcal{A}$ succeeds in generating a Type B forgery. By definition, we have $\Pr[\mathsf{E}'_0] = \Pr[\mathsf{E}_{\mathrm{B}}]$.

Game_1**:** This game is the same as the previous game, but the challenger uses a truly random function instead of $\mathsf{PRF}(K, \cdot)$ to derive u for each signing query. In more details, the challenger prepares an empty list Q at the beginning of the game. Then, when $\mathcal{A}$ makes a signing query for $(\mathbf{x}, \Delta)$ such that $\Delta \notin T$, the challenger samples $u \xleftarrow{\$} \mathbb{Z}_p$ and use the value to generate the signature. The challenger also adds (Δ, g^u) to Q. At the end of the game, $\mathcal{A}$ outputs a forgery $(C^\star, \Delta^\star, z^\star, \sigma^\star)$. The challenger outputs 1 if and only if the forgery satisfies the winning condition of the single-shot unforgeability game and $(\Delta^\star, g^{u^\star}) \in Q$. By a straightforward reduction to the security of PRF, we have $\Pr[\mathsf{E}'_1] \geq \Pr[\mathsf{E}'_0] - \mathsf{negl}(\kappa)$.

Game_2**:** In this game, we change the way the challenger samples u in order to answer signing queries. Given a signing query $(\mathbf{x}, \Delta)$, the challenger first chooses $\tilde{u} \xleftarrow{\$} \mathbb{Z}_p$ and defines u as

$$u = \tilde{u} + \sum_{i \in [n]} y_i \cdot \left(a^{m+1}/b_i c_i\right).$$

Then, $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}})$ is returned to $\mathcal{A}$. We can see that this change is only conceptual since the joint distribution of $(u, \tilde{u})$ is unchanged. More specifically, u is distributed uniformly at random over $\mathbb{Z}_p$ and $\tilde{u}$ satisfy Eq. (10) in both games. We therefore have $\Pr[\mathsf{E}'_2] = \Pr[\mathsf{E}'_1]$.

Game_3**:** In this game, the challenger randomly chooses $i^\star \xleftarrow{\$} [q_s]$ at the beginning of the game, where q_s is the upper-bound on the number of signing queries made by $\mathcal{A}$. Then, at the end of the game, the challenger checks whether the $i^\star$-th signing query made by $\mathcal{A}$ is of the form $(\Delta, \mathbf{x})$ such that $\Delta = \Delta^\star$. If this does not hold, the challenger aborts without outputting anything. Otherwise, the challenger outputs the same bit as the previous game. Since $i^\star$ is independent from the view of $\mathcal{A}$, we have $\Pr[\mathsf{E}'_3] = \Pr[\mathsf{E}'_2]/q_s$.

In order to finish the proof of the lemma, it remains to prove $\Pr[\mathsf{E}'_3] \leq \mathsf{negl}(\kappa)$. To do so, we replace the challenger in Game_3 with an adversary $\mathcal{B}$ against the partial adaptive unforgeability of our single-data HomSig scheme with advantage $\Pr[\mathsf{E}'_3]$. The adversary $\mathcal{B}$ proceeds as follows.

At the beginning of the game, $\mathcal{B}$ is given $(\{g^{a^j}\}_j, \{g^{c_i}\}_i, \{g^{a^j/b_i}\}_{i,j,j\neq m+1}, \{g^{a^{m+1}c_{i'}/b_i c_i}\}_{i,i',i\neq i'}, \{g^{ac_i}\}_i, \{g^{a^j/b_i c_i}\}_{i,j}, \{g^{a^j c_{i'}/b_i}\}_{i,i',j})$ as the partial verification key of the single-data HomSig scheme. Then, $\mathcal{B}$ runs $(\mathsf{DS.vk}, \mathsf{DS.sk}) \xleftarrow{\$} \mathsf{DS.KeyGen}(1^\kappa)$ and sets the verification key $\mathsf{HS.vk}$ of the multi-data HomSig scheme as Eq. (9). Then, $\mathcal{B}$ gives $\mathsf{HS.vk}$ to $\mathcal{A}$. $\mathcal{B}$ then prepares empty sets T and Q and chooses $i^\star \xleftarrow{\$} [q_s]$.

During the game, $\mathcal{A}$ makes signing queries. $\mathcal{B}$ answers the queries as follows.

- For the i-th query $(\Delta, \mathbf{x})$ such that $i \neq i^\star$, $\mathcal{B}$ returns $\perp$ if $\Delta \notin T$. Otherwise, it first chooses $\tilde{u} \xleftarrow{\$} \mathbb{Z}_q$, computes $\mathbf{y} = \mathsf{EncInp}(\mathbf{x})$, and sets
$$g^u := g^{\tilde{u}} \cdot \prod_{i \in [n]} \left(g^{a^{m+1}/b_i c_i}\right)^{y_i},$$
where the terms $\{g^{a^{m+1}/b_i c_i}\}$ are taken from the partial verification key. Then it runs $\mathsf{DS.Sign}(\mathsf{DS.sk}, g^u \| \Delta) \to \sigma_{\mathsf{DS}}$ and returns $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}})$ to $\mathcal{A}$. It also adds (Δ, g^u) and Δ to Q and T, respectively.
- For the $i^\star$-th query $(\Delta, \mathbf{x})$, $\mathcal{B}$ returns $\perp$ if $\Delta \in T$. Otherwise, $\mathcal{B}$ submits $\mathbf{x}$ to its challenger as target and is given g^u. Then, $\mathcal{B}$ makes a signing query for its challenger and is given $\tilde{u}$. Finally, it runs $\mathsf{DS.Sign}(\mathsf{DS.sk}, g^u \| \Delta) \to \sigma_{\mathsf{DS}}$ and returns $\boldsymbol{\sigma} = (g^u, \tilde{u}, \sigma_{\mathsf{DS}})$ to $\mathcal{A}$. It also adds (Δ, g^u) and Δ to Q and T, respectively.

Finally, $\mathcal{A}$ outputs a forgery $(C^\star, \Delta^\star, z^\star, \sigma^\star = (K_1^\star, K_2^\star, K_3^\star, g^{u^\star}, \sigma_{\mathsf{DS}}^\star, z^\star))$. $\mathcal{B}$ checks whether $C^\star \in \mathcal{C}$, $\sigma^\star \in \Sigma_{\mathsf{Evaled}}$, $\mathsf{HS.VerifyEvaled}(\mathsf{vk}, \Delta^\star, C^\star, z^\star, \sigma^\star) = \top$ and aborts if any of them does not hold. Otherwise, it also checks whether

the $i^\star$-th signing query was of the form $(\Delta, \mathbf{x})$ such that $\Delta = \Delta^\star$, $(\Delta^\star, g^{u^\star}) \in Q$, and $C^\star(\mathbf{x}) \neq z^\star$. If they do not hold, it aborts. Otherwise, it outputs $(K_1^\star, K_2^\star, K_3^\star, z^\star)$ as its forgery.

It is easy to see that $\mathcal{B}$ outputs a forgery iff E_3' occurs. Furthermore, $\mathcal{B}$ perfectly simulates Game_3. Therefore, assuming the partial adaptive unforgeability of the single-data scheme, we have $\Pr[\mathsf{E}_3'] \leq \mathsf{negl}(\kappa)$. □

This completes the proof of the theorem. □

Contents