Compact NIZKs from Standard Assumptions on Bilinear Maps

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

[1]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{shuichi.katsumata,yamada-shota}@aist.go.jp
[2]NTT Secure Platform Laboratories, Tokyo, Japan
{ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp

May 12, 2020

Abstract

A non-interactive zero-knowledge (NIZK) protocol enables a prover to convince a verifier of the truth of a statement without leaking any other information by sending a single message. The main focus of this work is on exploring short pairing-based NIZKs for all **NP** languages based on standard assumptions. In this regime, the seminal work of Groth, Ostrovsky, and Sahai (J.ACM'12) (GOS-NIZK) is still considered to be the state-of-the-art. Although fairly efficient, one drawback of GOS-NIZK is that the proof size is *multiplicative* in the circuit size computing the **NP** relation. That is, the proof size grows by $O(|C|\kappa)$, where C is the circuit for the **NP** relation and κ is the security parameter. By now, there have been numerous follow-up works focusing on shortening the proof size of pairing-based NIZKs, however, thus far, all works come at the cost of relying either on a non-standard knowledge-type assumption or a non-static q-type assumption. Specifically, improving the proof size of the original GOS-NIZK under the same standard assumption has remained as an open problem.

Our main result is a construction of a pairing-based NIZK for all of **NP** whose proof size is *additive* in $|C|$, that is, the proof size only grows by $|C| + \mathsf{poly}(\kappa)$, based on the decisional linear (DLIN) assumption. Since the DLIN assumption is the same assumption underlying GOS-NIZK, our NIZK is a strict improvement on their proof size.

As by-products of our main result, we also obtain the following two results: (1) We construct a *perfectly zero-knowledge* NIZK (NIPZK) for **NP** relations computable in $\mathbf{NC}^1$ with proof size $|w| \cdot \mathsf{poly}(\kappa)$ where $|w|$ is the witness length based on the DLIN assumption. This is the first pairing-based NIPZK for a non-trivial class of **NP** languages whose proof size is independent of $|C|$ based on a standard assumption. (2) We construct a universally composable (UC) NIZK for **NP** relations computable in $\mathbf{NC}^1$ in the erasure-free adaptive setting whose proof size is $|w| \cdot \mathsf{poly}(\kappa)$ from the DLIN assumption. This is an improvement over the recent result of Katsumata, Nishimaki, Yamada, and Yamakawa (CRYPTO'19), which gave a similar result based on a non-static q-type assumption.

The main building block for all of our NIZKs is a constrained signature scheme with *decomposable online-offline efficiency*. This is a property which we newly introduce in this paper and construct from the DLIN assumption. We believe this construction is of an independent interest.

1 Introduction

1.1 Background

Zero-knowledge proof system [GMR89] is an interactive protocol that allows a prover to convince a verifier about the validity of a statement without revealing anything beyond the fact that the statement is true. A variant of this, which is both practically and theoretically important, are *non-interactive* zero-knowledge (NIZK) proofs[1] [BFM88] where the prover is only required to send one message to the verifier to prove the validity of the statement in question. Not only have NIZKs shown to be a ubiquitous building block for cryptographic primitives and protocols, but it has also shown to be a mine of theoretical questions with interesting technical challenges.

[1] In the introduction, we do not distinguish between proofs and arguments for simplicity.

Unfortunately, it is known that NIZKs for non-trivial languages (i.e., **NP**) do not exist in the plain model where there is no trusted setup [GO94]. Therefore, NIZKs for non-trivial languages are typically constructed in the common reference string (CRS) model where the prover and verifier have access to a CRS generated by a trusted entity. We will call such NIZKs in the CRS model simply as NIZKs.

The most successful NIZK for all of **NP** is arguably the pairing-based NIZK of Groth, Ostrovsky, and Sahai [GOS12] (GOS-NIZK). GOS-NIZKs are based on the standard decisional linear (DLIN) or the subgroup decision (SD) assumptions. Due to its simplicity and efficiency, pairing-based NIZKs have flourished into a research topic on its own, and the original GOS-NIZK has been followed by many subsequent works trying to improve on it through various approaches. For example, many works such as [GS12, JR17, JR14, KW15] aim to make GOS-NIZK more efficient by limiting the language to very specific pairing induced languages, while other works such as [Gro10b, Lip12, GGPR13, DFGK14, Gro16] aim to gain efficiency by relying on a much stronger assumption known as knowledge assumptions (i.e., a type of non-falsifiable [Nao03, GW11] assumption). In fact, all works that achieve any notion of "better efficiency" compared to GOS-NIZK only succeeds by either restricting the language or by resorting to use stronger assumptions compared to DLIN or SD.

Similarly with many prior works, the main focus of "efficiency" in our work will be the *proof size* of the NIZK. Denoting C as the circuit computing the **NP** relation, GOS-NIZK requires a proof size as large as $O(|C|\kappa)$, where κ is the security parameter. Borrowing terminology from the recent work of Katsumata et al. [KNYY19a, KNYY19b], what we would like instead is a more *compact* proof size, that is, a proof size with only an additive overhead $|C| + \mathsf{poly}(\kappa)$ rather than a multiplicative overhead. For instance, the above latter approach using knowledge assumptions are known to achieve pairing-based NIZKs for **NP** with a significantly short proof size that only depends on the security parameter; in particular, the proof size does not even depend on the witness size. However, unfortunately, it is known that NIZKs with such an unusually short proof (i.e., proof size $\mathsf{poly}(\kappa) \cdot (|x| + |w|)^{o(1)}$ where x is the statement and w is the witness) inevitably require strong non-falsifiable assumptions [GW11]. The most compact NIZK based on any falsifiable assumption is due to [Gen09, GGI+15] which achieves proof size $|w| + \mathsf{poly}(\kappa)$. However, since it uses (circular secure) fully homomorphic encryption (FHE) its instantiation is solely limited to lattice-based assumptions. Other than lattice-based constructions, Groth [Gro10a] proposed a NIZK based on the security of Naccache-Stern public key encryption scheme [NS98] with a proof size $|C| \cdot \mathsf{polylog}(\kappa)$, which is asymptotically shorter than that of GOS-NIZK. Very recently, Katsumata et al. [KNYY19b] provided the first compact NIZK based on any falsifiable pairing-based assumption achieving a proof size of $|C| + \mathsf{poly}(\kappa)$. Their construction relies on a new primitive called homomorphic equivocal commitment (HEC), and they instantiate HEC using a *non-static* Diffie-Hellman type assumption recently introduced in [KNYY19a]. Unfortunately, the construction of HEC seems to be tailored to their specific non-static assumption, and it seems quite difficult to construct HEC based on a clean static assumption such as DLIN.

In summary, despite the considerable work that has been put into paring-based NIZKs, improving the proof size of GOS-NIZK while simultaneously maintaining the language and assumption has shown to be elusive. Therefore, in this work, the main question we ask is:

*Can we construct compact NIZKs for all of **NP** based on standard assumptions over a pairing group?*

1.2 Our Result

In this work, we present the first compact pairing-based NIZK for all of **NP** with proof size $|C| + \mathsf{poly}(\kappa)$ based on the DLIN assumption.[2] Along the way, we also obtain several interesting compact variants of our NIZK such as non-interactive *perfect* zero-knowledge (NIPZK) and universally composable NIZK (UC-NIZK) [GOS12] from the DLIN assumption. We provide a list of NIZKs which we achieve below and refer to Table 1 and 2 for comparison between prior works. We note that the table only includes NIZKs for **NP** based on falsifiable assumptions.

1. We construct a compact NIZK for all of **NP** languages with proof size $|C| + \mathsf{poly}(\kappa)$ based on the DLIN assumption.[3] This is the first NIZK to achieve a proof size shorter than that of GOS-NIZK under the same assumption required by GOS-NIZK. Moreover, if we assume the **NP** relation to be computable in $\mathbf{NC}^1$, the proof

[2] More precisely, we can base it on the weaker MDDH assumption, which includes the DLIN and symmetric external Diffie-Hellman (SXDH) assumptions as a special case.

[3] Strictly speaking, we also need to assume the CDH assumption in a subgroup of $\mathbb{Z}_p^*$ for a prime p. See Remark 5.5 for details.

size can be made as small as $|w| + \mathsf{poly}(\kappa)$, which matches the state-of-the-art of compact NIZKs from any primitive based on (possibly non-pairing) falsifiable assumptions, e.g., fully-homomorphic encryption [GGI+15]. Our NIZK can also be seen as an improvement of the recently proposed compact NIZK of Katsumata et al. [KNYY19b] in the following two aspects. First, our construction relies on a standard assumption, whereas theirs rely on a non-static q-type assumption. Second, our construction is fairly efficient since we only use pairing group operations in a black-box manner, whereas their construction is highly inefficient since they require pairing group operations in a non-black-box way.

2. We construct NIPZKs for **NP** languages that are computable in $\mathbf{NC}^1$ with proof size $|w| \cdot \mathsf{poly}(\kappa)$ from the DLIN assumption. This is the first pairing-based perfectly zero-knowledge NIZK for a non-trivial class of **NP** languages whose proof size is independent of $|C|$ based on a standard assumption.

3. We construct UC-NIZKs for **NP** languages that are computable in $\mathbf{NC}^1$ with proof size $|w| \cdot \mathsf{poly}(\kappa)$ from the DLIN assumption. This is an improvement over the recent result of Katsumata et al. [KNYY19b], which gave a similar result based on a non-static q-type assumption.

The main building block for all of our NIZKs is a constrained signature scheme with *decomposable online-offline efficiency*. This is a property which we newly introduce in this paper and construct from the DLIN assumption. We believe this construction is of independent interest.

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

Reference	Soundness	ZK	CRS size	Proof size	Assumption	Misc
[FLS99]	stat.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	$\mathsf{poly}(\kappa, \|C\|)$	trapdoor permutation†	
[Gro10a]	stat.	comp.	$\|C\| \cdot k_{\mathsf{tpm}} \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	$\|C\| \cdot k_{\mathsf{tpm}} \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	trapdoor permutation†	
[Gro10a]	stat.	comp.	$\|C\| \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	$\|C\| \cdot \mathsf{polylog}(\kappa) + \mathsf{poly}(\kappa)$	Naccache-Stern PKE	
[GOS12]	perf.	comp.	$\mathsf{poly}(\kappa)$	$O(\|C\|\kappa)$	DLIN/SD	
[GOS12]	comp.	perf.	$\mathsf{poly}(\kappa)$	$O(\|C\|\kappa)$	DLIN/SD	
[CHK07, Abu13]	stat.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	$\mathsf{poly}(\kappa, \|C\|)$	CDH	pairing group
[GGI+15]	stat.	comp.	$\mathsf{poly}(\kappa)$	$\|w\| + \mathsf{poly}(\kappa)$	FHE and CRS-NIZK	circular security
[KNYY19b]	comp.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	$\|C\| + \mathsf{poly}(\kappa)$	(n, m)-CDHER+CDH*	
[KNYY19b]	comp.	comp.	$\mathsf{poly}(\kappa, \|C\|, 2^d)$	$\|w\| + \mathsf{poly}(\kappa)$	(n, m)-CDHER+CDH*	limited to $\mathbf{NC}^1$ relation
[KNYY19b]	stat./comp.	comp.	$\mathsf{poly}(\kappa, \|x\|, \|w\|, d)$	$\mathsf{poly}(\kappa, \|x\|, \|w\|, d)$	LFE and CRS-NIZK	prover-efficient, implied by sub-exp. LWE
[KNYY19b]	stat./comp.	comp.	$(\|x\| + \|w\|) \cdot \mathsf{poly}(\kappa, d)$	$\tilde{O}(\|x\| + \|w\|) \cdot \mathsf{poly}(\kappa, d)$	LFE and CRS-NIZK‡	prover-efficient, implied by adaptive LWE
Section 5.1	comp.	comp.	$\mathsf{poly}(\kappa, \|C\|)$	$\|C\| + \mathsf{poly}(\kappa)$	DLIN +CDH*	
Section 5.1	comp.	comp.	$\mathsf{poly}(\kappa, \|C\|, 2^d)$	$\|w\| + \mathsf{poly}(\kappa)$	DLIN+CDH*	limited to $\mathbf{NC}^1$ relation
Section 5.2	comp.	perf.	$\mathsf{poly}(\kappa, \|C\|, 2^d)$	$\|w\| \cdot \mathsf{poly}(\kappa)$	DLIN	limited to $\mathbf{NC}^1$ relation

In column "Soundness" (resp. "ZK"), perf., stat., and comp. means perfect, statistical, and computational soundness (resp. zero-knowledge), respectively. In column "CRS size" and "Proof size", κ is the security parameter, $|x|, |w|$ is the statement and witness size, $|C|$ and d are the size and depth of the circuit computing the **NP** relation, and k_{tpm} is the length of the domain of the trapdoor permutation. In column "Assumption", (n, m)-CDHER stands for the (parameterized) computational DH exponent and ratio assumption, LFE stands for laconic functional evaluation, and sub-exp. LWE stands for sub-exponentially secure learning with errors (LWE).

* The CDH assumption should hold in a subgroup of $\mathbb{Z}_p^*$ for a prime p.

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

‡ We additionally require a mild assumption that the prover run time is linear in the size of the circuit computing the **NP** relation.

1.3 Technical Overview

1.3.1 Reviewing Previous Results

Here, we review definitions and previous results that are required for explaining our approach. We remark that we explain previous works [KW18, KNYY19a, KNYY19b] in terms of constrained signatures (CS) instead of homomorphic signatures, even though they are based on the latter primitive. This is because these primitives are actually equivalent as shown by Tsabary [Tsa17] and explaining in this way allows us to ignore small differences between our approach and previous ones that stem from the syntactic difference between them.

Table 2: Comparison of UC-NIZKs for **NP**.

Reference	Security (erasure-free)	CRS size	Proof size	Assumption	Misc
[GOS12]	adaptive (✓)	$\mathsf{poly}(\kappa)$	$O(\|C\|\kappa)$	DLIN/SD	
[GGI+15]	adaptive (✗)	$\mathsf{poly}(\kappa)$	$\|w\| + \mathsf{poly}(\kappa)$	FHE and UC-NIZK	circular security
[CsW19]	adaptive (✓)	$\mathsf{poly}(\kappa, d)$	$\|w\| \cdot \mathsf{poly}(\kappa, d)$	HTDF and UC-NIZK	
[KNYY19b]	adaptive (✓)	$\mathsf{poly}(\kappa, \|C\|, 2^d)$	$\|w\| \cdot \mathsf{poly}(\kappa)$	(n, m)-CDHER and UC-NIZK	limited to $\mathbf{NC}^1$ relation
Section 6	adaptive (✓)	$\mathsf{poly}(\kappa, \|C\|, 2^d)$	$\|w\| \cdot \mathsf{poly}(\kappa)$	DLIN	limited to $\mathbf{NC}^1$ relation

In column "CRS size" and "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. In column "Assumption", DLIN stands for the decisional linear assumption, SD stands for the subgroup decision assumption, HTDF stands for homomorphic trapdoor functions, and (n, m)-CDHER stands for the (parameterized) computational DH exponent and ratio assumption.

DP-NIZK and CS: We first explain the notion of designated prover NIZK (DP-NIZK), which is a relaxed notion of the standard notion of NIZK. In order to differentiate them, we call the latter CRS-NIZK in the following. In DP-NIZK, only a prover who possesses a secret proving key can generate a proof for an NP statement, and the verification can be done publicly by any entity. Here, the secret proving key is generated along with the CRS by a trusted entity. We require that soundness holds against a malicious prover who possesses the secret proving key and that zero-knowledge holds against a malicious verifier who only accesses the CRS and the proofs, but not the secret proving key. We then explain the notion of CS, which is a slightly simplified version of attribute-based signature [MPR11]. CS is an advanced form of signature where a signing key is associated with some circuit $C : \{0,1\}^\ell \to \{0,1\}$ and using the signing key, one can sign on a message x if $C(x) = 1$. The signature can be verified by a public verification key. As for security, we require unforgeability and privacy. The former requires that one cannot forge a valid signature on a message x if it only has a signing key $\mathsf{CS.sk}_C$ for C such that $C(x) = 0$. The latter requires that an honestly generated signature reveals nothing about the circuit C associated with the signing key that is used for generating the signature. In addition to the above security notions, we also require CS to have compact signatures in the sense that the size of the signatures is a fixed polynomial that is independent of the size of the circuit C and the length of the message x.

DP-NIZK from CS [KW18]: We then explain the generic construction of DP-NIZK from CS shown by Kim and Wu [KW18]. This will serve as a good starting point for us because their conversion allows us to convert a compact CS into a compact DP-NIZK as we will see. Let us fix an NP language L that is verified by a circuit R that takes as input a statement x and a witness w and outputs $R(x, w) \in \{0, 1\}$. In their construction, they set the CRS of the DP-NIZK to be a verification key of the CS. Furthermore, they set the secret proving key for the DP-NIZK to be a secret key K of an SKE and a CS signing key $\mathsf{CS.sk}_{C_K}$ for circuit C_K. Here, C_K is a circuit that takes as input an SKE ciphertext $\mathsf{SKE.ct}$ and a statement x and outputs 1 if $R(x, \mathsf{SKE.Dec}(K, \mathsf{SKE.ct})) = 1$ and 0 otherwise. To generate a proof for an NP statement x corresponding to a witness w, the prover encrypts the witness w by the SKE to obtain $\mathsf{SKE.ct} = \mathsf{SKE.Enc}(K, w)$ and then signs on the message $(x, \mathsf{SKE.ct})$ using the CS signing key for C_K. By the correctness of the SKE, we have $C_K(x, \mathsf{SKE.ct}) = R(x, w) = 1$, which implies the completeness of the DP-NIZK. The soundness of the protocol follows from the unforgeability of the underlying CS. This is because any valid proof for an invalid statement $x^* \notin L$ is a valid signature on $(x^*, \mathsf{SKE.ct}^*)$ for some $\mathsf{SKE.ct}^*$, for which we have $C_K(x^*, \mathsf{SKE.ct}^*) = R(x^*, \mathsf{SKE.Dec}(K, \mathsf{SKE.ct}^*)) = 0$. The zero-knowledge property of the protocol follows from the following intuition. From the privacy of the CS, information of K hardwired into the circuit C_K is not leaked from the CS signature. We, therefore, can use the security of SKE to conclude that $\mathsf{SKE.ct}$ leaks no information of the witness w.

We now focus on the efficiency of the resultant DP-NIZK. If we instantiate the DP-NIZK with an SKE with additive ciphertext overhead and a CS with compact signatures, this gives us a compact DP-NIZK. Note that an SKE scheme with additive ciphertext overhead can be realized from very mild assumptions such as CDH. Therefore, their result suggests that it suffices to construct compact CS in order to construct a compact DP-NIZK.

1.3.2 Overview of Our Approach

Here, we provide an overview of our approach. In high level, we follow the same approach as Katsumata et al. [KNYY19a, KNYY19b], who constructed a compact CRS-NIZK from a non-static assumption over bilinear maps.

Specifically, we will first construct a CS, then convert it into a DP-NIZK, and then modify it into a CRS-NIZK. However, our approach significantly differs from theirs in low level details. We will provide a comparison with their work after describing our approach in the following.

Compact DP-NIZK from a standard assumption: We set the construction of compact DP-NIZK from a static assumption as an intermediate goal. Thanks to the Kim-Wu conversion, the problem is reduced to the construction of a CS scheme with compact signatures from a static assumption. To achieve the goal, we follow the folklore conversion that converts an attribute-based encryption (ABE) into a CS that is somewhat reminiscent of the Naor conversion [BF01] (See e.g., [OT11]). In order to obtain the CS scheme with the desired properties, it turns out that we need to construct an adaptively secure ABE scheme whose ciphertext size is bounded by some fixed polynomial. Although there is no ABE scheme with the required properties from a static assumption in the literature, we are able to construct it by modifying the very recent ABE scheme proposed by Kowalczyk and Wee [KW19], who resolved the long-standing open problem of constructing adaptively secure ABE for $\mathbf{NC}^1$ whose ciphertext length is independent of the circuit size from a static assumption by cleverly adapting the piecewise guessing frameworks [FKPR14, FJP15, HJO+16, JW16, JKK+17, KW19] to the setting of ABE. We modify their scheme so that it has even shorter ciphertexts by aggregating the ciphertext components and adding extra components to the secret keys as was done in previous works on ABE with short ciphertexts [ALdP11, AHL+12, HW13]. The security proof for the scheme is again similar to that of Kowalczyk and Wee, where we decompose the secret keys into smaller pieces and gradually randomize them via carefully chosen sequence of hybrid games. The additional challenge for the proof in our setting is to deal with the extra components in the secret keys. We handle this by observing that the originally proof strategy by Kowalczyk and Wee for randomizing the secret keys works even with these extra components. From this ABE scheme, we can obtain a CS scheme with the desired properties. Furthermore, by applying the Kim-Wu conversion to the CS scheme, we obtain a new compact DP-NIZK from a static assumption. Although this is not our main goal, we note that this improves the compact DP-NIZK scheme from a non-static assumption by Katsumata et al. [KNYY19a].

Removing Secret Proving Key: We then try to remove the necessity of the secret proving key from the DP-NIZK described above to obtain a CRS-NIZK. Toward this goal, our first idea is to make the signing key of the CS scheme public by including it into the CRS. When we do so, we stop hardwiring the secret key K of the SKE into the circuit associated with the signing key and change the circuit so that it takes K as an input. The obvious reason for this is because we would like to use the security of SKE at some later point. More concretely, we include $\mathsf{CS.sk}_C$ into the CRS, where C is a circuit that takes as input the secret key K of SKE, a statement x, and a ciphertext $\mathsf{SKE.ct}$ of SKE and outputs $R(x, \mathsf{SKE.Dec}(K, \mathsf{SKE.ct}))$. When generating a proof, the prover chooses a random K on its own, computes $\mathsf{SKE.ct} \xleftarrow{\$} \mathsf{SKE.Enc}(K, w)$, and signs on the message $(x, \mathsf{SKE.ct}, K)$ by using $\mathsf{CS.sk}_C$ to obtain a signature $\mathsf{CS}.\sigma$, which is possible because we have $C(x, \mathsf{SKE.ct}, K) = 1$ by the definition of C. The problem with this approach is that we do not know what components to publish as the final proof. More specifically, we run into the following deadlock: If we include K into the proof, then the scheme is not zero-knowledge anymore because one can decrypt $\mathsf{SKE.ct}$ by using K to retrieve w. On the other hand, if we do not include K into the proof, we can no longer verify the validity of $\mathsf{CS}.\sigma$ since K, which is now a part of the message, is required to verify the signature.

Introducing Non-Compact NIZK: We resolve the above issue by using a CRS-NIZK that is not necessarily compact (non-compact NIZK in the following) and change the scheme so that it proves the validity of the CS signature without revealing K nor the signature. In more detail, the prover generates $K, \mathsf{SKE.ct} \xleftarrow{\$} \mathsf{SKE.Enc}(K, w), \mathsf{CS}.\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{CS.sk}_C, (x, \mathsf{SKE.ct}, K))$ as above. It then proves that there exists $(K, \mathsf{CS}.\sigma)$ such that $\mathsf{CS}.\sigma$ is a valid signature on a message $(x, \mathsf{SKE.ct}, K)$ under the verification key $\mathsf{CS.vk}$ by using the non-compact NIZK. It then outputs $(\mathsf{SKE.ct}, \mathsf{CS}.\sigma, \pi)$ as the final proof, where π is the non-compact proof for the above statement.

We then explain that the scheme satisfies soundness and zero-knowledge. To see this, we first observe that to break the soundness of the resultant NIZK scheme, it is necessary to break the soundness of the underlying non-compact NIZK or generate a valid CS signature on $(x^*, \mathsf{SKE.ct}^*, K^*)$ such that $x^* \notin L$. By our assumption, the former is impossible. Furthermore, the latter is also impossible, since we have $C(x^*, \mathsf{SKE.ct}^*, K^*) = 0$ for any choice of K^* and $\mathsf{SKE.ct}^*$ and thus it implies a forgery against the CS scheme. The zero-knowledge property of the scheme holds since the proof consists of the SKE ciphertext and the proof of the non-compact NIZK. Intuitively, since the latter does not leak the information about K, we can use the security of SKE to conclude that w is hidden from the adversary.

While this gives a secure construction, it is unclear whether this is a step forward at this point since we merely constructed a NIZK from a CS by further assuming a NIZK, which seems to be a vacuous statement. Furthermore,

the construction we described so far is not compact since the relation proven by the underlying non-compact NIZK is verified by a circuit whose size depends on $|C|$. To see this, we recall that the verification circuit for the relation proven by the non-compact NIZK takes as input the statement $x' = (\mathsf{CS.vk}, x, \mathsf{SKE.ct})$ and witness $w' = (K, \mathsf{CS}.\sigma)$ and outputs 1 if and only if $\mathsf{CS}.\sigma$ is a valid signature on $(x, \mathsf{SKE.ct}, K)$. This circuit is not compact, since it takes as input x, which can be as large as $|C|$ in general and $\mathsf{CS.vk}$, which is much larger than $|C|$ in our specific CS scheme.

Exploiting the Special Efficiency Property of the CS: We observe that what should be kept secret in the above construction are K and $\mathsf{CS}.\sigma$,[4] and $(x, \mathsf{SKE.ct})$ can be made public without losing the zero-knowledge property. To get a clearer understanding of the problem, we slightly generalize and simplify the problem as follows. What we would like to do is to give a compact proof that we have a valid signature $\mathsf{CS}.\sigma$ on a message (y, z) for public y and secret z without revealing z nor $\mathsf{CS}.\sigma$ using a non-compact NIZK. Here, y is not compact while z and $\mathsf{CS}.\sigma$ are compact. In our context, $y = (x, \mathsf{SKE.ct})$ and $z = K$. In this generalized setting, the above approach is equivalent to proving that $\mathsf{CS}.\sigma$ is a valid signature on (y, z) under the verification key $\mathsf{CS.vk}$. This relation is verified by a circuit that directly takes $(\mathsf{CS.vk}, (y, z), \mathsf{CS}.\sigma)$ as inputs. This approach does not work simply because the input is not compact.

Our first observation is that if we were somehow able to compress the verification circuit size of the relation proven by the non-compact NIZK to be a fixed polynomial without changing the functionality, then the resultant NIZK scheme will have compact proofs. Fortunately, our CS scheme has a nice property that brings us closer to this goal. Namely, in the scheme, the verifier can aggregate the verification key $\mathsf{CS.vk}$ depending on a message m to obtain an aggregated verification key $\mathsf{CS.vk}_m$, which is of fixed polynomial size. Then, a signature $\mathsf{CS}.\sigma$ can be verified by using *only* the aggregated verification key $\mathsf{CS.vk}_m$. In particular, the verification circuit no longer takes m as an input. Typically, the aggregation of the verification key is done offline, where one is allowed to perform heavy computation, and the actual verification step is done online, where the computation is very fast even if m is a very long string. We call this property *online-offline efficiency*. We note that our CS scheme inherits this property from the underlying ABE scheme, where secret keys can be aggregated depending on an attribute in offline phase so that the decryption of a ciphertext corresponding to the same attribute in the online phase is very fast.

A natural approach to compress the verification circuit (for the non-compact NIZK) would be to replace the inputs $\mathsf{CS.vk}$ and (y, z) with its aggregated version $\mathsf{CS.vk}_{(y,z)}$. In particular, we replace the verification circuit which takes as input $\mathsf{CS.vk}$, (y, z), and $\mathsf{CS}.\sigma$ and verifies the signature with the corresponding online verification circuit which takes $\mathsf{CS.vk}_{(y,z)}$ and $\mathsf{CS}.\sigma$ as inputs. This circuit is compact thanks to the online-offline efficiency of the CS. However, since $\mathsf{CS.vk}_{(y,z)}$ cannot be publicly computed, we would have to move the term $\mathsf{CS.vk}_{(y,z)}$ into the witness. Furthermore, we additionally have to prove that $\mathsf{CS.vk}_{(y,z)}$ is honestly computed from $\mathsf{CS.vk}$ and (y, z) using the non-compact NIZK. The problem is that the resulting proof is not compact since this is a statement that involves non-compact terms. Put differently, even though we can compactly prove that we have a signature that passes the online verification under a compressed verification key, we cannot compactly prove that we honestly execute the offline phase to compute the compressed verification key.

As we saw above, the idea of compressing $\mathsf{CS.vk}$ depending on the entire string (y, z) does not work. Our idea is to "partially" compress $\mathsf{CS.vk}$ depending on the public part y and then use this compressed version of the verification key to construct the verification circuit for the non-compact NIZK. To enable the idea, let us assume that we can compress $\mathsf{CS.vk}$ with respect to a string y and obtain $\mathsf{CS.vk}_y$. Then, further assume that we can compress $\mathsf{CS.vk}_y$ into $\mathsf{CS.vk}_{(y,z)}$ using z, so that the verification of a message (y, z) is possible using $\mathsf{CS.vk}_{(y,z)}$. Furthermore, we require that the computational cost of compressing $\mathsf{CS.vk}_y$ into $\mathsf{CS.vk}_{(y,z)}$ depends only on $|z|$, not on $|y|$. Therefore if z is compact, we can compute $\mathsf{CS.vk}_{(y,z)}$ from $\mathsf{CS.vk}_y$ and z by a compact circuit. Assuming this property, we can solve the above generalized problem as follows: We first compress $\mathsf{CS.vk}$ depending on y to obtain $\mathsf{CS.vk}_y$. We then prove that there exists $\mathsf{CS}.\sigma$ and z such that $\mathsf{CS}.\sigma$ is a valid signature under $\mathsf{CS.vk}_{(y,z)}$, where $\mathsf{CS.vk}_{(y,z)}$ is obtained by compressing $\mathsf{CS.vk}_y$ depending on the string z. This statement can be proven compactly, since both verification under the verification key $\mathsf{CS.vk}_{(y,z)}$ and the compression of $\mathsf{CS.vk}_y$ into $\mathsf{CS.vk}_{(y,z)}$ can be done compactly. Furthermore, unlike the previous attempt, we do not have to prove that we honestly executed the offline computation. Namely, we do not have to prove the consistency between $\mathsf{CS.vk}$, y, and $\mathsf{CS.vk}_y$, since $\mathsf{CS.vk}_y$ is publicly computable from $\mathsf{CS.vk}$ and y. Therefore, it suffices to show that our CS scheme has the structure that allows one to compress the verification key in two steps. We name this property *online-offline decomposability* and show that our construction indeed has the property.[5]

[4] Note that $\mathsf{CS}.\sigma$ should be kept secret since it reveals partial information of K.

[5] Actually, the definition of online-offline decomposability is slightly different from the one in the main body, but the latter implies the former.

1.3.3 Comparison with Katsumata et al. [KNYY19b]

Here, we compare our approach with the one by Katsumata et al. [KNYY19a, KNYY19b], who showed a similar result from a non-static assumption. As we already mentioned, at the highest level, their approach is the same as ours in that they first construct a CS [KNYY19a], then convert it into a DP-NIZK, and then modify it into a CRS-NIZK [KNYY19b]. However, the way they obtained the CS, and the way they modify their DP-NIZK into a CRS-NIZK is significantly different from ours. We elaborate on this below.

Compact CS Scheme by Katsumata et al. [KNYY19a]: Similarly to us, their approach is to construct an ABE scheme and then convert it into a CS scheme. However, the requirements for the ABE are different from ours. For the ABE scheme, they require short secret keys, whereas we require short ciphertexts. Furthermore, they require the ABE scheme to be secure following a so-called "single-shot" reduction, where the reduction algorithm runs the adversary only once and perfectly simulates the view of the game. Roughly, this is equivalent to saying that the proof cannot go through hybrid arguments. Therefore, their approach does not seem to be promising when we try to construct a compact CS scheme from a *static* assumption. Notably, their single-shot reduction requirement excludes the dual system encryption methodology [Wat09], which is a powerful tool for proving the security of an ABE scheme from static assumptions. On the other hand, we manage to employ the dual system encryption methodology to obtain an ABE scheme with the desired properties from static assumptions.

From DP-NIZK to CRS-NIZK in Katsumata et al. [KNYY19b]: They construct a DP-NIZK (as an intermediate goal) by applying the Kim-Wu conversion on their CS scheme. They then modify their DP-NIZK to a CRS-NIZK scheme by a non-generic technique. Here, we review their approach and compare it with ours. Recall that, in general, a DP-NIZK constructed from a CS via the Kim-Wu conversion, the CRS consists of the verification key of the CS $\mathsf{CS.vk}$, and the secret proving key consists of the secret key of an SKE K and a signing key of the CS $\mathsf{CS.sk}_{C_K}$. Their observation was that they can divide the CS verification key $\mathsf{CS.vk}$ into two components $\mathsf{CS.vk} := (\mathsf{CS.vk}_0, \mathsf{CSvk}_1)$ such that $\mathsf{CS.vk}_1$ is very short and anyone can compute $\mathsf{CS.vk}_1$ from $\mathsf{CS.sk}_{C_K}$ and K. Note that as a stand-alone CS scheme, the secret key $\mathsf{CS.sk}_{C_K}$ is computed using the master key of the CS only *after* $\mathsf{CS.vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$ is defined. What they observe is that the other direction of the computation is possible using the specific structure of their CS scheme. In order to construct a CRS-NIZK using this special structure, they remove $\mathsf{CS.vk}_1$ from the CRS. Then they let the prover pick K and $\mathsf{CS.sk}_{C_K}$ on their own and let it compute $\mathsf{CS.vk}_1$. At this point, the prover can generate a proof as in the original DP-NIZK. In order to prevent the adversary to maliciously choose K, $\mathsf{CS.sk}_{C_K}$, and $\mathsf{CS.vk}_1$, they let the prover prove consistency among the components using a non-compact NIZK and outputs the proof along with $\mathsf{CS.vk}_1$. The additional consistency proof by the non-compact NIZK as well as $\mathsf{CS.vk}_1$ appended to the final proof does not harm the compactness of the resulting NIZK, since all parameters involved are compact.

We note that their approach is not applicable to our specific CS scheme. The reason is that our signing key for the CS is as large as the circuit size and we cannot prove the consistency between K, $\mathsf{CS.sk}_{C_K}$, and $\mathsf{CS.vk}_1$ compactly no matter how we divide the CS verification key. We, therefore, take a different path from theirs and this entails several challenges that are not present in their approach.

1.4 Related Work

The first NIZK for **NP** was given by [BFM88] based on the quadratic residuosity assumption and Feige et al. [FLS99] weakened the assumption to the existence of trapdoor permutations (whose arguments were later refined by several works [BY96, Gol04]). The next generation of NIZK following a completely different set of approaches were provided by Groth, Ostrovsky, and Sahai [GOS12] (GOS-NIZK) based on pairings. Due to its simplicity and efficiency, pairing-based NIZKs have flourished into a research topic on its own, and the original GOS-NIZK has been followed by many subsequent works [Gro10a, Gro10b, Lip12, GS12, GGPR13]. More than roughly a decade later, a new type of NIZKs based on indistinguishable obfuscation (iO) were proposed [SW14, BP15, BPW16, CL18]. Finally, very recently, a different path for designing NIZKs based on correlation intractable hash functions (CIH) [KRR17, CCRR18, CCH+19] have gained much attention and has finally lead to the closing of a long-standing problem of constructing NIZKs based on lattice-based assumptions [PS19].

2 Definitions

2.1 Preliminaries on Bilinear Maps

A bilinear group generator GGen takes as input 1^κ and outputs a group description $\mathbb{G} = (p, G_1, G_2, G_T, e, g_1, g_2)$, where p is a prime such that $p > 2^{2\kappa}$, G_1, G_2, and G_T are cyclic groups of order q, $e : G_1 \times G_2 \to G_T$ is a non-degenerate bilinear map, and g_1 and g_2 are generators of G_1 and G_2, respectively. We require that the group operations in G_1, G_2, and G_T as well as the bilinear map e can be efficiently computed. We employ the implicit representation of group elements: for a matrix $\mathbf{A}$ over $\mathbb{Z}_q$, we define $[\mathbf{A}]_1 := g_1^{\mathbf{A}}$, $[\mathbf{A}]_2 := g_2^{\mathbf{A}}$, $[\mathbf{A}]_T := g_T^{\mathbf{A}}$, where exponentiation is carried out component-wise.

Definition 2.1 (MDDH_k **assumption [EHK+13]).** *Let* GGen *be a group generator. We say that the matrix DDH* (MDDH_k) *assumption holds on* G_1 *with respect to* GGen*, if for all PPT adversaries* $\mathcal{A}$*, we have*

$$\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{Ms}]_1) \to 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{u}]_1) \to 1]|$$

is negligible, where the probability is taken over the choice of $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$, $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times k}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$, *and* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. *We can similarly define* MDDH_k *assumption on* G_2.

In fact, the above assumption is called MDDH_k assumption for uniform distribution by Escala et al. [EHK+13] since $\mathbf{M}$ is chosen uniformly at random. As shown by them, MDDH_k assumptions for uniform distribution is weaker than MDDH_k assumption for all other distributions and in particular is implied by the k-LIN assumption.

2.2 Symmetric Key Encryption

Here, we define symmetric key encryption (SKE). Our definition of SKE is taken from [KNYY19b]. In this definition, we explicitly introduce a setup algorithm that generates a public parameter. If we just think about an SKE scheme alone, then such a setup algorithm is redundant since we can think of a public parameter as part of a secret key. On the other hand, we need an SKE scheme with a short secret key size (among other properties) in the construction of our compact NIZK. For achieving such a short secret key, we explicitly introduce the setup algorithm so that a secret key need not include values that can be made public. We also note that we only require one-time security instead of CPA-security.

Let $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of message space. In the following, we occasionally drop the subscript and simply write $\mathcal{M}$ when the meaning is clear.

Definition 2.2 (Symmetric Key Encryption). *A symmetric key encryption (SKE) scheme* Π_{SKE} *for message space* $\mathcal{M}$ *consists of PPT algorithms* $(\mathsf{SKE.Setup}, \mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.

$\mathsf{SKE.Setup}(1^\kappa) \to \mathsf{pp}_{\mathsf{SKE}}$*: The setup algorithm takes as input the security parameter* 1^κ *and outputs a public parameter* $\mathsf{pp}_{\mathsf{SKE}}$.

$\mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}}) \to \mathsf{K}_{\mathsf{SKE}}$*: The key generation algorithm takes as input a public parameter* $\mathsf{pp}_{\mathsf{SKE}}$ *and outputs a secret key* $\mathsf{K}_{\mathsf{SKE}}$.

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

$\mathsf{SKE.Dec}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{K}_{\mathsf{SKE}}, \mathsf{ct}) \to \mathsf{M}$ *or* $\perp$*: The decryption algorithm takes as input a public parameter* $\mathsf{pp}_{\mathsf{SKE}}$*, 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. *For all* $\kappa \in \mathbb{N}$, $\mathsf{M} \in \mathcal{M}$, $\mathsf{pp}_{\mathsf{SKE}} \in \mathsf{SKE.Setup}(1^\kappa)$, *and* $\mathsf{K}_{\mathsf{SKE}} \in \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$, *we have* $\mathsf{SKE.Dec}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{K}_{\mathsf{SKE}}, \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{K}_{\mathsf{SKE}}, \mathsf{M})) = \mathsf{M}$.

One-Time Security. *For all PPT adversaries* $\mathcal{A}$ *and* $(\mathsf{M}_0, \mathsf{M}_1) \in \mathcal{M}^2$*, if we run* $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$, $\mathsf{K}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$, *and* $\mathsf{ct}_b \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{K}_{\mathsf{SKE}}, \mathsf{M}_b)$ *for* $b \in \{0, 1\}$*, then we have*

$$|\Pr[\mathcal{A}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{M}_0) = 1] - \Pr[\mathcal{A}(\mathsf{pp}_{\mathsf{SKE}}, \mathsf{M}_1) = 1]| = \mathsf{negl}(\kappa).$$

For our construction of NIZKs in the following sections, we require an SKE scheme whose ciphertext overhead (i.e., $|\mathsf{ct}| - |m|$) and secret key length are $\mathsf{poly}(\kappa)$ and whose decryption algorithm can be represented as a circuit in $\mathbf{NC}^1$. Such an SKE exists under the CDH assumption in a subgroup of $\mathbb{Z}_p^*$ as shown in [KNYY19b].

Lemma 2.3. *If the CDH assumption holds in a subgroup of $\mathbb{Z}_p^*$, then for any $m = \mathsf{poly}(\kappa)$, there exists a one-time secure SKE scheme with message space $\{0,1\}^m$, ciphertext overhead $\mathsf{poly}(\kappa)$[6], and secret key length $\mathsf{poly}(\kappa)$ (independently of m) whose decryption algorithm can be represented as a circuit in $\mathbf{NC}^1$.*

2.3 One-Time Signature

Let $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of message space. In the following, we occasionally drop the subscript and simply write $\mathcal{M}$ when the meaning is clear.

Definition 2.4 (One-time Signature). *A one-time signature (OTS) scheme Π_{OTS} for message space $\mathcal{M}$ consists of PPT algorithms* $(\mathsf{OTS.KeyGen}, \mathsf{OTS.Sign}, \mathsf{OTS.Verify})$.

$\mathsf{OTS.KeyGen}(1^\kappa) \to (\mathsf{vk}_{\mathsf{OTS}}, \mathsf{sigk}_{\mathsf{OTS}})$*: The key generation algorithm takes as input the security parameter 1^κ and outputs a verification key $\mathsf{vk}_{\mathsf{OTS}}$ and a signing key $\mathsf{sigk}_{\mathsf{OTS}}$.*

$\mathsf{OTS.Sign}(\mathsf{sigk}_{\mathsf{OTS}}, \mathsf{M}) \to \sigma$*: The signing algorithm takes as input a signing key $\mathsf{sigk}_{\mathsf{OTS}}$ and a message $\mathsf{M} \in \mathcal{M}$ and outputs a signature σ.*

$\mathsf{OTS.Verify}(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{M}, \sigma) \to \top$ *or* $\bot$*: The decryption algorithm takes as input a verification key $\mathsf{vk}_{\mathsf{OTS}}$, a message M and a signature σ and outputs $\top$ to indicate acceptance of the signature and $\bot$ otherwise.*

Correctness. *For all $\kappa \in \mathbb{N}$, $\mathsf{M} \in \mathcal{M}$, $(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{sigk}_{\mathsf{OTS}}) \in \mathsf{OTS.KeyGen}(1^\kappa)$, and $\sigma \in \mathsf{OTS.Sign}(\mathsf{sigk}_{\mathsf{OTS}}, \mathsf{M})$, we have $\mathsf{OTS.Verify}(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{M}, \sigma) = \top$.*

Strong One-Time Unforgeability. *For all $\kappa \in \mathbb{N}$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,we have*

$$\Pr\left[\begin{array}{r|rl} \mathsf{OTS.Verify}(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{M}^*, \sigma^*) = \top & (\mathsf{vk}_{\mathsf{OTS}}, \mathsf{sigk}_{\mathsf{OTS}}) & \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^\kappa) \\ (\mathsf{M}, \sigma) \neq (\mathsf{M}^*, \sigma^*) & (\mathsf{M}, \mathsf{st}) & \xleftarrow{\$} \mathcal{A}_1(\mathsf{vk}_{\mathsf{OTS}}), \\ & \sigma & \xleftarrow{\$} \mathsf{OTS.Sign}(\mathsf{sigk}_{\mathsf{OTS}}, \mathsf{M}) \\ & (\mathsf{M}^*, \sigma^*) & \xleftarrow{\$} \mathcal{A}_2(\mathsf{st}, \sigma) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

2.4 Non-Interactive Zero-Knowledge Arguments

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 $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. Below, we define non-interactive zero-knowledge arguments for **NP** languages.[7]

Definition 2.5 (NIZK Arguments). *A non-interactive zero-knowledge (NIZK) argument Π_{NIZK} for the relation $\mathcal{R}$ consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$.

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

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

[6] In fact, the SKE scheme achieves a ciphertext length that equals the message length. However, any SKE scheme with a fixed-polynomial additive ciphertext overhead and secret key length suffice for our work.

[7] We say it is a non-interactive zero-knowledge *proofs* when the soundness property holds for even unbounded adversaries. In this paper, we will only be interested in computationally bounded adversaries.

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

We consider the following requirements for a NIZK argument Π_{NIZK}*, where the probabilities are taken over the random choice of the algorithms.*

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

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

Adaptive Soundness. *For all PPT adversaries* $\mathcal{A}$*, if we run* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$*, then we have*

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

Non-Adaptive Soundness. *We also consider the slightly weaker variant of adaptive soundness above. For all PPT adversaries* $\mathcal{A}$ *and for all* $x \notin \mathcal{L}$*, if we run* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$*, then we have*

$$\Pr[\pi \xleftarrow{\$} \mathcal{A}(1^\kappa, \mathsf{crs}, x) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = \top] = \mathsf{negl}(\kappa).$$

Zero-Knowledge. *For all adversaries* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that if we run* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$ *and* $(\overline{\mathsf{crs}}, \bar{\tau}) \xleftarrow{\$} \mathcal{S}_1(1^\kappa)$*, then we have*

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

where $\mathcal{O}_0(\mathsf{crs}, x, w)$ *outputs* $\mathsf{Prove}(\mathsf{crs}, x, w)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise, and* $\mathcal{O}_1(\overline{\mathsf{crs}}, \bar{\tau}, x, w)$ *outputs* $\mathcal{S}_2(\overline{\mathsf{crs}}, \bar{\tau}, x)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise. We say it is* computational (resp. statistical) *zero-knowledge if the adversary is computationally bounded (resp. unbounded). Moreover, we further say it is* perfect *zero-knowledge if the above r.h.s. equals 0 for computationally unbounded adversaries.*

We also define a stronger notion of soundness called *extractability* following [KNYY19b].

Definition 2.6 (Extractability). *An NIZK argument is said to be extractable if the following is satisfied:*
Extractability. *There is a deterministic algorithm* Extract *(called extractor) such that for all PPT adversary* $\mathcal{A}$*, we have*

$$\Pr\left[\begin{array}{r|l} \mathsf{Verify}(\mathsf{crs}, x, \pi) = \top & \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa), (x, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}), \\ (x, w) \notin \mathcal{R} & w \xleftarrow{\$} \mathsf{Extract}(r_{\mathsf{Setup}}, \pi) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

where r_{Setup} *is the randomness used in* Setup *to generate* crs*.*

We can convert any adaptively sound NIZK into an extractable one additionally assuming the existence of PKE [KNYY19b].

Lemma 2.7. *If there exist an adaptively sound NIZK for all of* ***NP*** *and a CPA-secure PKE scheme, then there exists an extractable NIZK for all of* ***NP****.*

2.5 Key-Policy Attribute-Based Encryption

Here, we define key-policy attribute-based encryption (KP-ABE) for monotone Boolean formulae.

Definition 2.8 (Attribute-Based Encryption). *A key-policy attribute-based encryption (KP-ABE) scheme with attribute space* $\{0,1\}^n$ *for family of Boolean formulae* $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}\}$ *consists of PPT algorithms* (Setup, KeyGen, Enc, Dec).

$\mathsf{Setup}(1^\kappa, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$*: The setup algorithm on input the security parameter* 1^λ *and the input length* 1^n*, outputs a master secret key* msk *and a master public key* mpk*.*

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$*: The key generation algorithm on input a master secret key* msk *and a monotone Boolean formula* $f \in \mathcal{F}$*, outputs a secret key* sk_f*.*

$\mathsf{Enc}(\mathsf{mpk}, x, M) \to \mathsf{ct}$*: The encryption algorithm on input the master public key* mpk*, an attribute* $x \in \{0,1\}^n$*, and a message* M *and outputs a ciphertext* ct_x*.*

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, \mathsf{ct}_x) \to M$ *or* $\perp$*: The decryption algorithm on input the verification key* mpk*, a secret key* sk_f*, and a ciphertext* ct_x *and outputs either message* M *or* $\perp$ *(indicating the ciphertext is valid).*

A KP-ABE scheme must satisfy the following requirements.

Correctness. For all $\kappa \in \mathbb{N}$, $n = n(\kappa) \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^n)$, $x \in \{0,1\}^n$, $f \in \mathcal{F}$ such that $f(x) = 1$, message M, and $\mathsf{sk}_f \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, f)$, we have

$$\Pr[\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, \mathsf{Enc}(\mathsf{mpk}, x, M)) = M] = 1.$$

Adaptive Security. We define (adaptive) security of ABE. The security notion is defined by the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger runs $(\mathsf{mpkmsk}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^n)$ and gives mpk to $\mathcal{A}$. It also prepares an empty list $\mathcal{Q}$.

Key Queries: $\mathcal{A}$ can adaptively make key queries unbounded polynomially many times throughout the game. When $\mathcal{A}$ queries $f \in \mathcal{F}$, the challenger runs $\mathsf{sk}_f \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}$. Finally, the challenger updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{f\}$.

Challenge Phase: At some point, $\mathcal{A}$ specifies its target attribute x^* and messages (M_0, M_1) it would like to be challenged on. Then, the challenger flips a random coin $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, encrypt the corresponding message $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, x, M_{\mathsf{coin}})$, and returns ct to $\mathcal{A}$.

Guess: Eventually, $\mathcal{A}$ outputs coin' as its guess for coin. We say $\mathcal{A}$ *wins* if $\mathsf{coin}' = \mathsf{coin}$. Furthermore, we say that $\mathcal{A}$ is *admissible* if $f(x^*) = 0$ holds for all $f \in \mathcal{Q}$ at the end of the game.

We say the ABE scheme is adaptively secure if the winning probability for all admissible PPT adversaries $\mathcal{A}$ in the above game is $\mathsf{negl}(\kappa)$, where the probability is taken over the randomness of all algorithms.

2.6 $\mathbf{NC}^1$ Circuits and Monotone Formulae

Here, we define Monotone Boolean formula following Kowalczyk and Wee [KW19].

Monotone Boolean formula. A monotone Boolean formula $f : \{0,1\}^n \to \{0,1\}$ is specified by a directed acyclic graph (DAG) with three kinds of nodes: input gate nodes, gate nodes, and a single output node. Input nodes have in-degree 0 and out-degree 1, AND/OR nodes have in-degree (fan-in) 2 and out-degree (fan-out) 1, and the output node has in-degree 1 and out-degree 0. We number the edges (wires) $1, 2, \ldots, m$, and each gate node is defined by a tuple (g, a_g, b_g, c_g) where $g : \{0,1\}^2 \to \{0,1\}$ is either AND or OR, a_g and b_g are the incoming wires, c_g is the outgoing wire and $a_g, b_g < c_g$. The size of a formula m is the number of edges in the underlying DAG and the depth of a formula d is the length of the longest path from the output node.

$\mathbf{NC}^1$ and Boolean formulae. The following lemma summarizes the well-known equivalence between the monotone formulae and $\mathbf{NC}^1$ circuits.

Lemma 2.9. *Let* $d = d(\kappa), n = n(\kappa)$*, and* $s = s(\kappa)$ *be integers. There exist integer parameters* $m = m(d, n, s)$ *and* deterministic *algorithms* EncInp *and* EncCir *with the following properties.*

- $\mathsf{EncInp}(x) \to \hat{x} \in \{0,1\}^{2n}$*, where* $x \in \{0,1\}^n$*.*

- $\mathsf{EncCir}(C) \to f$, *where* $C : \{0,1\}^n \to \{0,1\}$ *is a circuit with depth and size bounded by* d *and* s*, respectively and* f *is a monotone Boolean formula of size* m *with input space being* $\{0,1\}^{2n}$.

We have $f(\hat{x}) = 1$ *if and only if* $C(x) = 1$. *Furthermore, the running time of* EncCir *is* $\mathsf{poly}(n, s, 2^d)$. *In particular, if* C *is a polynomial-sized circuit with logarithmic depth (i.e., if the circuit is in* $\boldsymbol{NC}^1$*),* EncCir *runs in polynomial time and we have* $m = \mathsf{poly}(\kappa)$. *Furthermore, for* $x \in \{0,1\}^n$*, we have*

$$\hat{x} = x_1 \bar{x}_1 x_2 \bar{x}_2 \cdots x_n \bar{x}_n,$$

where $\bar{x}_i$ *is the flip of* x_i.

Sketch of the proof of Lemma 2.9. Since the above lemma is a restate of the well-known fact, we only provide a sketch of the proof here (See Section 2.1 of [KW19] and Section 2.2 of [GGH+13] for example). We describe the algorithm EncCir that converts C into an equivalent monotone formula f. Given input C, it first converts C into an equivalent Boolean formula f'', which is not necessarily monotone. This can be done with blowup in size at most 2^d. It then eliminates negation gates inside f'' (namely, all gates except for input gates) to obtain equivalent formula f' such that negation gates only appear at the input level. This can be done by repeatedly applying the De Morgan's rule to the gates in f'' from the output gate to the input gates. Finally, EncCir modifies f' to form a monotone formula f by changing the input gate of f' that reads the i-th bit (resp., negation of the i-th bit) of x to be a gate that reads the $2i$-th bit (resp., $2i-1$-th bit) of $\hat{x}$. It is not difficult to see that $f'(x) = f(\hat{x})$ holds for $\hat{x} = \mathsf{EncInp}(x)$. □

2.7 Piecewise Guessing Frameworks and Pebbling Games

Here, we review some ideas from piecewise guessing frameworks and pebbling games [FKPR14, FJP15, HJO+16, JW16, JKK+17, KW19]. These notions have been used to prove adaptive security of complex cryptographic objects that often involve circuits or graphs such as but not limited to garbled circuits [HJO+16, JW16, JKK+17] and ABE [KW19].

In the following, we follow the formalization by Kowalczyk and Wee [KW19]. We use $\langle \mathcal{A}, \mathsf{G} \rangle$ to denote the output of an adversary $\mathcal{A}$ in an interactive game G, and an adversary wins if the output is 1, so that the winning probability is denoted by $\Pr[\langle \mathcal{A}, \mathsf{G} \rangle = 1]$.

Piecewise Guessing Frameworks. Suppose we have two games G_0 and G_1 which we would like to show indistinguishable. In both games, the adversary $\mathcal{A}$ makes some adaptive choices that defines a string $z \in \{0,1\}^R$. In order to prove indistinguishability between the games, we introduce a family of hybrid games $\{\mathsf{H}^u\}_{u \in \{0,1\}^{R'}}$ where the behavior of the challenger in game H^u is specified by a string u. We also introduce functions $h_0, \ldots, h_L : \{0,1\}^R \to \{0,1\}^{R'}$ and games $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ for any $u_0, u_1 \in \{0,1\}^{R'}$, where

$\widehat{\mathsf{H}}_{\ell,b}(u_0, u_1)$ is the same as H^{u_b}, except that we replace the output with 0 whenever $(h_{\ell-1}(z), h_\ell(z)) \neq (u_0, u_1)$.

In this setting, Kowalczyk and Wee showed the following lemma.

Lemma 2.10 (Adaptive security lemma [KW19, Lemma 1]). *Fix* G_0, G_1 *along with* $h_0, \ldots, h_L : \{0,1\}^R \to \{0,1\}^{R'}$ *and* $\{\mathsf{H}^u\}_{u \in \{0,1\}^{R'}}$ *such that*

$$\forall z^* \in \{0,1\}^R : \mathsf{H}^{h_0(z^*)} = \mathsf{G}_0, \quad \mathsf{H}^{h_L(z^*)} = \mathsf{G}_1.$$

Suppose there exists an adversary $\mathcal{A}$ *such that*

$$\Pr[\langle \mathcal{A}, \mathsf{G}_0 \rangle = 1] - \Pr[\langle \mathcal{A}, \mathsf{G}_0 \rangle = 1] \geq \epsilon$$

then there exists $\ell \in [L]$ *and* $u_0, u_1 \in \{0,1\}^{R'}$ *such that*

$$\Pr[\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1) \rangle = 1 - \langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1) \rangle = 1] \geq \frac{\epsilon}{2^{2R'} L}.$$

Pebbling Games. Fix a formula $f : \{0,1\}^n \to \{0,1\}$ and an input $x \in \{0,1\}^n$ for which $f(x) = 0$. We are allowed to place or remove pebbles on nodes and gates in f, subject to the following rules. Formally, we consider the following rules.

Definition 2.11 (Pebbling Rules).

1. *We can place or remove a pebble on any AND gate for which (at least) one input wire comes out of a node with pebble on it.*
2. *We can place or remove a pebble on any OR gate for which all of the incoming wires come out of nodes which have pebbles on them.*
3. *We can place or remove a pebble on any input node for which $x_i = 0$.*

The goal of the pebbling game is to end up in a configuration where only the root gate has a pebble, starting from the configuration where there is no pebble at all on any gate, through a sequence of the pebbling instructions that abide by the above rules. Since the description size for an intermediate configuration and the number of pebbling instructions to reach the goal affect the security loss, we want them to be as small as possible. We introduce the following lemma, which was shown by Kowalczyk and Wee who improves the bound on the description size for a configuration compared to the similar results shown in [JKK+17].

Lemma 2.12 (Pebbling for NC^1 [KW19, Theorem 1]). *For every input $x \in \{0,1\}^n$ and any monotone formula f of depth d and fan-in two for which $f(x) = 0$, there exists a sequence of $L(d) = 8^d$ pebbling instructions such that every intermediate pebbling configurations can be described using $R'(d) = 3d$ bits. Furthermore, there exists an algorithm* Pebble *that takes as input f and x and outputs such a sequence in time* $\mathsf{poly}(2^d)$. *In particular,* $\mathsf{Pebble}(f, x)$ *runs in time* $\mathsf{poly}(\kappa)$ *when $d = O(\log \kappa)$.*

3 KP-ABE with Compact Ciphertexts

In this section, we give the construction of KP-ABE scheme for monotone Boolean formulae with constant-size ciphertexts by extending the scheme by Kowalczyk and Wee [KW19]. The scheme will be used in the construction of compact constrained signature scheme in Section 4, which will in turn be used for the construction of our compact NIZKs in Section 5. Our KP-ABE scheme would be of independent interest, since this is the first KP-ABE scheme for Boolean formulae with constant-size ciphertexts that is secure under a static assumption (rather than non-static q-type assumption).

3.1 Preliminaries

First, we review the secret sharing scheme for monotone Boolean formulae used by Kowalczyk and Wee, which is based on secret sharing schemes in [IK02, VNS+03, JKK+17].

Definition 3.1 (Secret Sharing). *A secret sharing scheme consists of two algorithms* $(\mathsf{share}, \mathsf{reconstruct})$.

$\mathsf{share}(f, \mu)$**:** *This algorithm takes a (monotone) Boolean formula $f : \{0,1\}^n \to \{0,1\}$ and $\mu \in \mathbb{Z}_p$ and outputs shares $\mu_1, \ldots, \mu_{\hat{m}} \in \mathbb{Z}_p$ and a function $\rho : [\hat{m}] \to \{0, 1, \ldots, n\}$. We assume that ρ is deterministically determined from f.*

$\mathsf{reconstruct}(f, x, \{\mu_j\}_{j\in S})$**:** *This algorithms takes an input $x \in \{0,1\}^n$ for f, f, and a subset of shares $\{\mu_j\}_{j\in S}$ where $S \subseteq [\hat{m}]$ and outputs the original value μ.*

A secret sharing scheme satisfies the following properties.

Correctness: *For all $x \in \{0,1\}^n$, $f : \{0,1\}^n \to \{0,1\}$, $\mu \in \mathbb{Z}_p$, $(\{\mu_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, \mu)$ such that $f(x) = 1$, it holds that* $\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0\vee x_{\rho(j)}=1}) = \mu$.

Security: *For all $x \in \{0,1\}^n$, $f : \{0,1\}^n \to \{0,1\}$, $\mu, \mu' \in \mathbb{Z}_p$ such that $f(x) = 0$, the following distributions are the same:*

$$\{\{\mu_j\}_{\rho(j)=0\vee x_{\rho(j)}=1} \mid (\{\mu_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, \mu)\} \equiv \{\{\mu'_j\}_{\rho(j)=0\vee x_{\rho(j)}=1} \mid (\{\mu'_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, \mu')\}$$

Linearity: *The algorithm* $\mathsf{reconstruct}$ *is a linear function of the shares over $\mathbb{Z}_p$. That is, there exists $\omega_j \in \mathbb{Z}_p$ for $j \in [\hat{m}]$ and we can compute $\mu = \sum_{\rho(j)=0\vee x_{\rho(j)}=1} \omega_j \mu_j$.*

We present their secret sharing scheme $(\mathsf{share}, \mathsf{reconstruct})$ in Figure 1 as it is. The scheme satisfies Definition 3.1. As Kowalczyk and Wee observed, it is easy to extend the secret sharing scheme to treat vectors of secrets. That is, for a vector $\mathbf{v} \in \mathbb{Z}_p^k$, we define

$$\mathsf{share}(f, \mathbf{v}) := (\{\mathbf{v}_j = (v_{1,j}, \ldots, v_{k,j})\}_{j\in[\hat{m}]}, \rho) \quad \text{where } (\{v_{i,j}\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, v_i)$$

and

$$\mathsf{reconstruct}(f, x, \{\mathbf{v}_j\}_{\rho(j)=0\vee x_{\rho(j)}=1}) := \sum_{\rho(j)=0\vee x_{\rho(j)}=1} \omega_j \mathbf{v}_j \quad \text{where } \{\omega_j\}_{j\in[\hat{m}]} \text{ is defined as above.}$$

$\underline{\mathsf{share}(f, \mu)}$

Input: A formula $f : \{0,1\}^n \to \{0,1\}$ of size m (that is, the number of edges in f is m) and a secret $\mu \in \mathbb{Z}_p$.

1. For each non-output wire $j = 1, \ldots, m-1$, choose a uniformly random $\hat{\mu}_j \xleftarrow{\$} \mathbb{Z}_p$. For the output wire, set $\hat{\mu}_m := \mu$.
2. For each outgoing wire j from input node i, add $\mu_j := \hat{\mu}_j$ to the output set of shares and set $\rho(j) := i$.
3. For each AND gate g with input wires a, b and output wire c, add $\mu_c := \hat{\mu}_c + \hat{\mu}_a + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c) := 0$.
4. For each OR gate g with input wires a, b and output wire c, add $\mu_{c_a} := \hat{\mu}_c + \hat{\mu}_a \in \mathbb{Z}_p$ and $\mu_{c_b} := \hat{\mu}_c + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c_a) := 0$ and $\rho(c_b) := 0$.
5. Output $(\{\mu_j\}_{j\in[\hat{m}]}, \rho)$.

$\underline{\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0\vee x_{\rho(j)}=1})}$

Input: A formula $f : \{0,1\}^n \to \{0,1\}$ of size m, $x \in \{0,1\}^n$, and $\{\mu_j\}_{\rho(j)=0\vee x_{\rho(j)}=1}$.

From the leaves of the formula to the root, we compute the output wire value $\hat{\mu}_c$ at each node.

1. Given $\hat{\mu}_a, \hat{\mu}_b$ associated with the input wires a and b of an AND gate, we compute $\hat{\mu}_c = \mu_c - \hat{\mu}_a - \hat{\mu}_b$.
2. Given $\hat{\mu}_a$ (or $\hat{\mu}_b$) associated with the input wires a (or b) of an OR gate, we compute $\hat{\mu}_c = \mu_{c_a} - \hat{\mu}_a$ (or $\hat{\mu}_c = \mu_{c_b} - \hat{\mu}_b$).

Output $\mu = \hat{\mu}_m$.

Figure 1: Information-theoretic linear secret sharing for monotone Boolean formulae by Kowalczyk and Wee [KW19]

3.2 Construction

Here, we give the construction of KP-ABE with short ciphertext from the MDDH_k assumption.

$\mathsf{Setup}(1^\kappa, 1^n)$**:** Run $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}$, $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times k}$ for $i \in [n]$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and output

$$\mathsf{mpk} = ([\mathbf{A}]_1, [\mathbf{A}\mathbf{W}_1]_1, \ldots, [\mathbf{A}\mathbf{W}_n]_1, e([\mathbf{A}]_1, [\mathbf{v}]_2)), \quad \mathsf{msk} = (\mathbf{v}, \mathbf{W}_1, \ldots, \mathbf{W}_n).$$

$\mathsf{Enc}(\mathsf{mpk}, x, M)$**:** To encrypt a message $M \in G_T$ for a string $x \in \{0,1\}^n$, sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and output

$$\mathsf{ct}_x = \left(\mathsf{ct}_1 := [\mathbf{s}^\top \mathbf{A}]_1, \quad \mathsf{ct}_2 = \left[\mathbf{s}^\top \sum_{i:x_i=1} \mathbf{A}\mathbf{W}_i\right]_1, \quad \mathsf{ct}_3 := e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2) \cdot M\right).$$

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** To generate a secret key for a Boolean formula f, sample $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \mathbf{v})$, $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_p^k$ and output sk_f, which consists of the following.

$$\left(\left\{\mathsf{sk}_j := [\mathbf{r}_j]_2,\ \mathsf{sk}_{\rho(j),j} := [\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2,\ \{\mathsf{sk}_{i,j} := [\mathbf{W}_i \mathbf{r}_j]_2\ \}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right)$$

where $\mathbf{W}_0 = \mathbf{0}$ and $\hat{m}$ is the number of shares. We note that for j such that $\rho(j) = 0$, we have $[n]\setminus\{\rho(j)\} = [n]$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, \mathsf{ct}_{\mathbf{x}})$: Compute ω_j such that $\mathbf{v} = \sum_{j:\rho(j)=0\vee x_{\rho(j)}=1} \omega_j \mathbf{v}_j$ and output

$$\mathsf{ct}_3 \cdot e\left(\mathsf{ct}_2, \prod_{j:\rho(j)=0\vee x_{\rho(j)}=1} \mathsf{sk}_j^{\omega_j}\right) \cdot e\left(\mathsf{ct}_1, \prod_{j:\rho(j)=0\vee x_{\rho(j)}=1} \left(\prod_{i:x_i=1} \mathsf{sk}_{i,j}\right)^{\omega_j}\right)^{-1}.$$

Correctness. The correctness follows since we have

$$\prod_{j:\rho(j)=0\vee x_{\rho(j)}=1} \left(\prod_{i:x_i=1} \mathsf{sk}_{i,j}\right)^{\omega_j} = \left[\mathbf{v} + \sum_{i:\hat{x}_i=1} \mathbf{W}_i \mathbf{r}\right]_2,$$

$$\prod_{j:\rho(j)=0\vee x_{\rho(j)=1}} \mathsf{sk}_j^{\omega_j} = [\mathbf{r}]_2, \qquad \text{where} \quad \mathbf{r} = \sum_{j:\rho(j)=0\vee \hat{x}_{\rho(j)}=1} \omega_j \mathbf{r}_j$$

for honestly generated secret key sk for f such that $f(x) = 1$ from the correctness of the secret sharing.

3.3 Key Lemma for Security Proof

In this section, we prove the following lemma, which will be used for proving the security of our KP-ABE scheme.

Lemma 3.2. *Under the* MDDH_k *assumption, we have*

$$\left|\Pr\left[\begin{array}{c}\mu^{(0)}, \mu^{(1)} \xleftarrow{\$} \mathbb{Z}_p; \mathbf{w}_0 := \mathbf{0}, \mathbf{w}_1, \ldots, \mathbf{w}_n \xleftarrow{\$} \mathbb{Z}_p^k; \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{F},0}(\cdot), \mathcal{O}_{\mathsf{X}}(\cdot), \mathcal{O}_{\mathsf{E}}(\cdot)}(\mu^{(0)})\end{array}\right] - \Pr\left[\begin{array}{c}\mu^{(0)}, \mu^{(1)} \xleftarrow{\$} \mathbb{Z}_p; \mathbf{w}_0 := \mathbf{0}, \mathbf{w}_1, \ldots, \mathbf{w}_n \xleftarrow{\$} \mathbb{Z}_p^k; \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{F},1}(\cdot), \mathcal{O}_{\mathsf{X}}(\cdot), \mathcal{O}_{\mathsf{E}}(\cdot,\cdot)}(\mu^{(0)})\end{array}\right]\right| = \mathsf{negl}(\kappa),$$

where $\mathcal{A}$ *adaptively interacts with three oracles:*

$$\mathcal{O}_{\mathsf{F},\beta}(f) := \left(\{\mu_j\}_{j:\rho(j)=0} \cup \left\{[\mathbf{r}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}^\top \mathbf{r}_j]_2, \left\{[\mathbf{w}_i^\top \mathbf{r}_j]_2\right\}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right) \text{ where } (\{\mu_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, \mu^{(\beta)})$$

$$\mathcal{O}_{\mathsf{X}}(x) := (\{\mathbf{w}_i\}_{i:x_i=1})$$

$$\mathcal{O}_{\mathsf{E}}() := \left([\mathbf{r}]_2, \left\{[\mathbf{w}_i^\top \mathbf{r}]_2\right\}_{i\in[n]}\right) \text{ where } \mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$$

with the restriction that (i) only one query is made to each of $\mathcal{O}_{\mathsf{F},\beta}(\cdot)$ *and* $\mathcal{O}_{\mathsf{X}}(\cdot)$*, and (ii) the queries* f *and* x *to* $\mathcal{O}_{\mathsf{F},\beta}(\cdot)$ *and* $\mathcal{O}_{\mathsf{X}}(\cdot)$ *respectively, satisfy* $f(x) = 0$*.*

Note that the statement of the lemma is similar to that of Theorem 2 in [KW19]. There, $\mathcal{O}_{\mathsf{F},\beta}(f)$ returns

$$\left(\{\mu_j\}_{j:\rho(j)=0} \cup \left\{[\mathbf{r}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}\mathbf{r}_j]_2\right\}_{j:\rho(j)\neq 0}\right)$$

and $\mathcal{O}_{\mathsf{E}}$ takes as input $i \in [m]$ and returns $([\mathbf{r}]_2, [\mathbf{w}_i^\top \mathbf{r}]_2)$.[8] Since the answers by the oracles in [KW19] can be simulated by our oracles by just stripping off appropriate components, our statement is stronger than theirs. Nonetheless, we can prove the above lemma with very similar proof to that of Theorem 2 in [KW19].

We define

$$\Pr[\langle \mathcal{A}, \mathsf{G}_\beta^{\text{1-abe}}\rangle = 1] := \Pr\left[\begin{array}{c}\mu^{(0)}, \mu^{(1)} \xleftarrow{\$} \mathbb{Z}_p; \mathbf{w}_0 := \mathbf{0}, \mathbf{w}_1, \ldots, \mathbf{w}_n \xleftarrow{\$} \mathbb{Z}_p^k; \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{F},\beta}(\cdot), \mathcal{O}_{\mathsf{X}}(\cdot), \mathcal{O}_{\mathsf{E}}(\cdot)}(\mu^{(0)})\end{array}\right].$$

Then, our goal is to prove $\left|\Pr[\langle \mathcal{A}, \mathsf{G}_0^{\text{1-abe}}\rangle = 1] - \Pr[\langle \mathcal{A}, \mathsf{G}_1^{\text{1-abe}}\rangle = 1]\right| = \mathsf{negl}(\kappa)$. To do so, we define a modified secret sharing scheme share^u in Figure 2 and hybrid distributions in Definition 3.3. In $\mathsf{share}^u(f, \mu)$, we first generate shares of μ as in share. However, we then replace all shares that are associated with output wires of the pebbled gates with random values.

[8] More accurately, $\mathcal{O}_{\mathsf{E}}$ takes as input $[M]_2 \in G_2$ in addition to i in [KW19]. But we can ignore the additional input $[M]_2$ without loss of generality.

$\mathsf{share}^u(f, \mu)$

Input: A formula $f : \{0,1\}^n \to \{0,1\}$, a secret $\mu \in \mathbb{Z}_p$, and a pebbling configuration u of the nodes of f.

1. Compute $(\{\mu'_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}(f, \mu)$ as defined in Figure 1.
2. For each μ'_j, if $j \in u$ (i.e., if j is the output wire of a pebbled node), then choose $\mu_j \xleftarrow{\$} \mathbb{Z}_p$. Otherwise, set $\mu_j := \mu'_j$.
3. Output $(\{\mu_j\}_{j\in[\hat{m}]}, \rho)$.

Figure 2: Pebbling-modified secret sharing scheme by Kowalczyk and Wee [KW19]

Definition 3.3 (hybrid distribution and function taken from [KW19]). *Let H^u be the same as $\mathsf{G}_0^{\text{1-abe}}$ except that we use $\mathsf{share}^u(f, \mu^{(0)})$ in the implementation of oracle $\mathcal{O}_{\mathsf{F},0}(\cdot)$. Let $h_\ell : \boldsymbol{NC}^1 \times \{0,1\}^n \to \{0,1\}^{R'}$ be as follows: h_ℓ takes as input formula f and input $x \in \{0,1\}^n$ where $f(x) = 0$ and outputs the pebbling configuration created from following the first ℓ instructions from $\mathsf{Pebble}(f, x)$ defined in Lemma 2.12.*

From the definition, when $\ell = 0$, h_0 is a constant function for all f and x since there is no pebble in the configuration. When $\ell = L$, h_L is also a constant function for all f and x where $f(x) = 0$ since the pebbling strategy in Lemma 2.12 yields a configuration with single pebble on the root gate.

By the description of share^u, it holds that for all such f and x,

- $\mathsf{H}^{h_0(f,x)} \equiv \mathsf{G}_0^{\text{1-abe}}$ since $\mathsf{share}^{h_0(f,x)}(f, \mu^{(0)}) = \mathsf{share}(f, \mu^{(0)})$.
- $\mathsf{H}^{h_L(f,x)} \equiv \mathsf{G}_1^{\text{1-abe}}$ since $\mathsf{share}^{h_L(f,x)}(f, \mu^{(0)}) = \mathsf{share}(f, \mu^{(1)})$ for an independently random $\mu^{(1)}$.

As in Section 2.7, we define games $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ for $\ell \in [0, L]$. For these games, we have the following lemma.

Lemma 3.4 (neighboring indistinguishability). *For all $\ell \in [L]$ and $u_0, u_1 \in \{0,1\}^{R'}$, it holds that*

$$\Pr[\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle = 1] - \Pr[\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle = 1] \le n \cdot \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B}}(\kappa).$$

Our proof strategy basically follows that of Kowalczyk and Wee. However, unlike the proofs of Kowalczyk and Wee [KW19], we directly prove Lemma 3.4 by using the MDDH_k assumption instead of using CPA-secure secret-key encryption based on the MDDH_k assumption. The reason is that $\mathcal{O}_{\mathsf{F},\beta}(f)$ outputs not only $\{\mu_j\}_{\rho(j)=0} \cup ([\mathbf{r}_j]_2, [\mu_j + \mathbf{w}^\top_{\rho(j)}\mathbf{r}_j]_2)$ but also $\left\{[\mathbf{w}_i^\top \mathbf{r}_j]_2\right\}_{i\in[n]\setminus\{\rho(j)\}}$, which shares the same randomness of $[\mu_j + \mathbf{w}^\top_{\rho(j)}\mathbf{r}_j]_2$ (This is the SKE part of Kowalczyk and Wee $\mathcal{O}_{\mathsf{F}}$). Therefore, we cannot use the abstraction of SKE.

Before we prove Lemma 3.4, we prove Lemma 3.2 by using Lemma 3.4.

Proof of Lemma 3.2. By Lemma 3.4 and adaptive security lemma (Lemma 2.10), we obtain

$$\Pr[\langle \mathcal{A}, \mathsf{G}_0^{\text{1-abe}}\rangle = 1] - \Pr[\langle \mathcal{A}, \mathsf{G}_1^{\text{1-abe}}\rangle = 1] \le 2^{2R'} \cdot L \cdot n \cdot \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B}}(\kappa)$$

where R' is the description size of a pebbling configuration and L is the length of the sequence of the pebbling instructions. By pebbling lemma (Lemma 2.12), we have $R' \le 3d$ and $L \le 8^d$ and thus

$$\Pr[\langle \mathcal{A}, \mathsf{G}_0^{\text{1-abe}}\rangle = 1] - \Pr[\langle \mathcal{A}, \mathsf{G}_1^{\text{1-abe}}\rangle = 1] \le 2^{6d} \cdot 8^d \cdot n \cdot \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B}}(\kappa).$$

Since $d = O(\log \kappa)$, this completes the proof of Lemma 3.2. □

Now, we move to the proof of Lemma 3.4.

Proof of Lemma 3.4. The only difference between $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ is that the former uses share^{u_0} in $\mathcal{O}_{\mathsf{F},0}$ and the latter uses share^{u_1} in $\mathcal{O}_{\mathsf{F},0}$. There are two cases to consider.

- There does not exist $x' \in \{0,1\}^n$ such that $h_{\ell-1}(f, x') = u_0 \wedge h_\ell(f, x') = u_1$.

- There exists $x' \in \{0,1\}^n$ such that $h_{\ell-1}(f,x') = u_0 \wedge h_\ell(f,x') = u_1$.

In the first case, both $\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0,u_1)\rangle$ and $\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0,u_1)\rangle$ abort with probability 1 and two games are perfectly indistinguishable.

Thus, we focus on the second case in the rest of the proof. In this case, u_0 and u_1 are neighboring pebbling configurations in $\mathsf{Pebble}(f,x')$. Therefore, the difference between the two games should be caused by an instruction that follows one of the pebbling rules in Definition 2.11. As the proof by Kowalczyk and Wee, we can consider three cases depending on which instruction changes the configuration u_0 into u_1. The second and third cases below are almost the same as those by Kowalczyk and Wee, but we provide the proofs for completeness.

Pebble/unpebble input node with out-going wire j^*: Let us consider the case where the difference between the games is caused by the third item of pebbling rules in Definition 2.11. In particular, let us assume that input node with out-going wire $j^* \in [\hat{m}]$ is unpebbled in $\widehat{\mathsf{H}}_{\ell,0}(u_0,u_1)$ and it is pebbled in $\widehat{\mathsf{H}}_{\ell,1}(u_0,u_1)$. (The proof for the other case is similar.) Note that such j^* is uniquely determined from u_0 and u_1. The only difference between the games is that $\mathsf{share}^{u_0}(f,\mu^{(0)})$ sets μ_{j^*} to be an honestly generated share of $\mu^{(0)}$, but $\mathsf{share}^{u_1}(f,\mu^{(0)})$ sets it to be a random element in $\mathbb{Z}_p$. We assume that $\mathcal{A}$ chooses x and f such that $x_{\rho(j^*)} = 0$ since otherwise both games output 0.

Claim 3.5. Under the MDDH_k assumption, for the case of pebble/unpebble input node, the two neighboring hybrids $\widehat{\mathsf{H}}_{\ell,0}(u_0,u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0,u_1)$ are indistinguishable.

Proof. For the sake of contradiction, let us assume that $\mathcal{A}$ distinguishes the neighboring games. We then construct an algorithm $\mathcal{B}$, which is given the MDDH_k instance $([\mathbf{M}]_2,[\mathbf{u}]_2) \in G_2^{(k+1)\times k} \times G_2^{k+1}$ and breaks the MDDH_k problem by using $\mathcal{A}$. That is, $\mathcal{B}$ tries to distinguish whether $[\mathbf{u}]_2 = [\mathbf{M}\mathbf{s}]_2$ for $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$ or $[\mathbf{u}]_2 \xleftarrow{\$} (G_2)^{k+1}$ by interacting with $\mathcal{A}$. In order to do so, $\mathcal{B}$ must simulate $\mathcal{O}_{\mathsf{F},0}(\cdot)$, $\mathcal{O}_{\mathsf{X}}(\cdot)$, and $\mathcal{O}_{\mathsf{E}}()$ for the adversary $\mathcal{A}$.

Setup. Given $([\mathbf{M}]_2,[\mathbf{u}]_2)$, $\mathcal{B}$ first picks random guess $i^* \xleftarrow{\$} [n]$ for $\rho(j^*)$, where $\rho : [\hat{m}] \to \{0,1,\dots,n\}$ is the function associated with the function f that will be chosen by $\mathcal{A}$ later in the game. $\mathcal{B}$ then chooses $\mathbf{w}_i \xleftarrow{\$} \mathbb{Z}_p^k$ for $i \in [n] \setminus \{i^*\}$ and $\mu^{(0)}, \mu^{(1)} \xleftarrow{\$} \mathbb{Z}_p$ and sends $\mu^{(0)}$ to $\mathcal{A}$.

Throughout the game, $\mathcal{B}$ implicitly sets

$$[\mathbf{r}_{j^*}]_2 \coloneqq [\overline{\mathbf{M}}\mathbf{s}]_2 \quad \text{and} \quad \mathbf{w}_{i^*} \coloneqq (\underline{\mathbf{M}} \cdot \overline{\mathbf{M}}^{-1})^\top,$$

where $\mathbf{r}_{j^*}$ is the randomness that will be used to answer the query to $\mathcal{O}_{\mathsf{F},0}$, and $\overline{\mathbf{M}} \in \mathbb{Z}_p^{k\times k}$ and $\underline{\mathbf{M}} \in \mathbb{Z}_p^{1\times k}$ are the first k rows and the last *row* of $\mathbf{M}$, respectively. That is, if $[\mathbf{u}]_2 = [\mathbf{M}\mathbf{s}]_2$, we have

$$[\mathbf{u}]_2 = \left[\begin{pmatrix} \mathbf{r}_{j^*} \\ \mathbf{w}_{i^*}^\top \mathbf{r}_{j^*} \end{pmatrix}\right]_2$$

since $\mathbf{w}_{i^*}^\top \mathbf{r}_{j^*} = (\underline{\mathbf{M}} \cdot \overline{\mathbf{M}}^{-1})\overline{\mathbf{M}}\mathbf{s} = \underline{\mathbf{M}}\mathbf{s}$. On the other hand, if $[\mathbf{u}]_2 \xleftarrow{\$} (G_2)^{k+1}$, we have $[\mathbf{u}]_2 = \left[\binom{\mathbf{r}_{j^*}}{u'}\right]_2$ where $\mathbf{r}_{j^*} \xleftarrow{\$} \mathbb{Z}_p^k$ and $u' \xleftarrow{\$} \mathbb{Z}_p$ are uniformly random.

Simulating $\mathcal{O}_{\mathsf{F},0}(\cdot)$. Given a query f to $\mathcal{O}_{\mathsf{F},0}(\cdot)$ made by $\mathcal{A}$, $\mathcal{B}$ first checks whether $i^* = \rho(j^*)$. If it does not hold, it outputs a random bit and aborts. Otherwise, $\mathcal{B}$ computes $(\{\mu_j\}_{j\in[\hat{m}]}, \rho) \leftarrow \mathsf{share}^{u_0}(f,\mu^{(0)})$, chooses $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [\hat{m}] \setminus \{j^*\}$, and returns

$$\begin{aligned}
&\{\mu_j\}_{j\in[\hat{m}]:\rho(j)=0} \\
&\cup \left\{ [\overline{\mathbf{M}}\mathbf{s}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}^\top \overline{\mathbf{M}}\mathbf{s}_j]_2, \left(\left\{ [\mathbf{w}_i^\top \overline{\mathbf{M}}\mathbf{s}_j]_2 \right\}_{i\in[n]\setminus\{\rho(j),i^*\}}, [\underline{\mathbf{M}}\mathbf{s}_j]_2 \right) \right\}_{j\in[\hat{m}]\setminus\{j^*\}:\rho(j)\neq i^*} \quad (1) \\
&\cup \left\{ [\overline{\mathbf{M}}\mathbf{s}_j]_2, [\mu_j + \underline{\mathbf{M}}\mathbf{s}_j]_2, \left\{ [\mathbf{w}_i^\top \overline{\mathbf{M}}\mathbf{s}_j]_2 \right\}_{i\in[n]\setminus\{i^*\}} \right\}_{j\in[\hat{m}]\setminus\{j^*\}:\rho(j)=i^*} \quad (2) \\
&\cup \left\{ [\overline{\mathbf{u}}]_2, [\mu_{j^*} + \underline{\mathbf{u}}]_2, \left\{ [\mathbf{w}_i^\top \overline{\mathbf{u}}]_2 \right\}_{i\in[n]\setminus\{i^*\}} \right\} \quad (3)
\end{aligned}$$

in the appropriate order, where $\overline{\mathbf{u}}$ and $\underline{\mathbf{u}}$ are the first k elements and the $(k+1)$-th element of $\mathbf{u}$, respectively. Note that the above terms can be efficiently computed since $\mathcal{B}$ has $\{\mathbf{w}_i\}_{i\in[n]\setminus\{i^*\}}$, $[\mathbf{M}]_2$, $[\mathbf{u}]_2$ and $\{\mathbf{s}_j\}_{j\in[m]\setminus\{j^*\}}$.

It is easy to see that $\{\mu_j\}_{j\in[\hat{m}]:\rho(j)=0}$ generated as above are distributed as in the games. We claim that the terms in Equation (1) and (2) are distributed as the terms in

$$\left\{[\mathbf{r}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}^\top \mathbf{r}_j]_2, \left\{[\mathbf{w}_i^\top \mathbf{r}_j]_2\right\}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]} \tag{4}$$

with corresponding j after the appropriate rearrangement of the terms, where $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_k$ for $j \in [\hat{m}]$ and $\{\mu_j\}_{j\in[\hat{m}]}$ are chosen as $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ in Equation (4).[9] We also claim that the terms in Equation (3) are distributed as the terms in Equation (4) with $j = j^*$ in $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ if $\mathbf{u} = \mathbf{M}\mathbf{s}$ and as in $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ if $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. We prove them in the following.

- We first focus on the terms in Equation (1). If we set $\mathbf{r}_j := \overline{\mathbf{M}}\mathbf{s}_j$, we have

  $$[\overline{\mathbf{M}}\mathbf{s}_j]_2 = [\mathbf{r}_j]_2 \quad \text{and} \quad [\underline{\mathbf{M}}\mathbf{s}_j]_2 = [(\underline{\mathbf{M}} \cdot \overline{\mathbf{M}}^{-1})(\overline{\mathbf{M}}\mathbf{s}_j)]_2 = [\mathbf{w}_{\rho(j^*)}^\top \mathbf{r}_j]_2.$$

  We can see that $\mathbf{r}_j$ is distributed uniformly at random over $\mathbb{Z}_q^k$ since so is $\mathbf{s}_j$ and $\overline{\mathbf{M}}$ is invertible. Therefore, the distribution of the terms in Equation (1) is the same as those in Equation (4) for $j \in [\hat{m}] \setminus \{j^*\}$ such that $\rho(j) \neq i^*$.
- We then focus on the terms in Equation (2). Similarly to the case above, by setting $\mathbf{r}_j := \overline{\mathbf{M}}\mathbf{s}_j$, we can see that the distribution of the terms in Equation (2) is the same as those in Equation (4) for $j \in [\hat{m}] \setminus \{j^*\}$ such that $\rho(j) = i^*$.
- We finally focus on the terms in Equation (3). If $[\mathbf{u}]_2 = [\mathbf{M}\mathbf{s}]_2$, then $\mathcal{B}$ perfectly simulates $\mathcal{O}_{\mathsf{F},0}(f)$ since $[\mathbf{u}]_2 = \left[\begin{pmatrix} \mathbf{r}_{j^*} \\ \mathbf{w}_{i^*}^\top \mathbf{r}_{j^*} \end{pmatrix}\right]_2$ in this case. That is, the terms in Equation (3) can be represented as

  $$([\mathbf{r}_{j^*}]_2, [\mu_{j^*} + \mathbf{w}_{i^*}^\top \mathbf{r}_{j^*}]_2, \{[\mathbf{w}_i^\top \mathbf{r}_{j^*}]_2\}_{i\in[n]\setminus\{i^*\}}),$$

  where we recall that $i^* = \rho(j^*)$. If $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, then $[\mu_{j^*} + \underline{\mathbf{u}}]_2$ and $[\overline{\mathbf{u}}]_2 = [\mathbf{r}_{j^*}]_2$ are uniformly random over G_2 and G_2^k, respectively. Thus, $\mathcal{B}$ perfectly simulates the game where we use $\mathsf{share}^{u_1}(f, \mu^{(0)})$.

Simulating $\mathcal{O}_{\mathsf{X}}(\cdot)$. Given a query x to $\mathcal{O}_{\mathsf{X}}(\cdot)$ made by $\mathcal{A}$, $\mathcal{B}$ aborts and outputs a random bit if $x_{i^*} = 1$. Otherwise, $\mathcal{B}$ sends $\{\mathbf{w}_i\}_{i:x_i=1}$ to $\mathcal{A}$. Note that although $\mathcal{B}$ does not have $\mathbf{w}_{i^*}$, it can answer the query as long as the guess i^* for $\rho(j^*)$ is correct, since we have $x_{i^*} = x_{\rho(j^*)} = 0$ in that case.

Simulating $\mathcal{O}_{\mathsf{E}}()$. For a query to $\mathcal{O}_{\mathsf{E}}()$, $\mathcal{B}$ chooses $\mathbf{s}' \xleftarrow{\$} \mathbb{Z}_p^k$ and computes $[\mathbf{M}\mathbf{s}']_2$ (we can compute this from $[\mathbf{M}]_2$ and $\mathbf{s}'$). It then sets $[\mathbf{r}']_2 := [\overline{\mathbf{M}}\mathbf{s}']_2$ and returns

$$\left([\mathbf{r}']_2, \{[\mathbf{w}_i^\top \mathbf{r}']_2\}_{i\in[n]\setminus\{i^*\}}, [\underline{\mathbf{M}}\mathbf{s}']_2\right)$$

in the appropriate order. We can see $[\underline{\mathbf{M}}\mathbf{s}']_2 = [(\underline{\mathbf{M}} \cdot \overline{\mathbf{M}}^{-1})(\overline{\mathbf{M}}\mathbf{s}')]_2 = [\mathbf{w}_{i^*}^\top \mathbf{r}']_2$ since $\mathbf{w}_{i^*} = (\underline{\mathbf{M}} \cdot \overline{\mathbf{M}}^{-1})^\top$. Therefore, $\mathcal{B}$ perfectly simulates $\mathcal{O}_{\mathsf{E}}()$.

Guess. When $\mathcal{A}$ halts and outputs a bit, $\mathcal{B}$ outputs the same bit as its guess.

Based on the argument above, $\mathcal{B}$ perfectly simulates $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ if $[\mathbf{u}]_2 = [\mathbf{M}\mathbf{s}]_2$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ if $[\mathbf{u}]_2 \xleftarrow{\$} G_2^{k+1}$ for $\mathcal{A}$ conditioned that the guess i^* for $\rho(j^*)$ is correct. Since i^* is chosen uniformly at random and hidden from the adversary, the guess is correct with probability $1/n$. Therefore, if $\mathcal{A}$ has distinguishing advantage ϵ for these two games, $\mathcal{B}$ has advantage ϵ/n against the MDDH_k problem. This completes the proof of the claim. □

[9] Notice that the distribution of $\{\mu_j\}_{j\neq j^*}$ in $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ is the same as that in $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$.

Pebble/unpebble AND gate with out-going wire c and input wires a, b corresponding to nodes g_a, g_b: The difference between the two games is that $\mathsf{share}^{u_0}(f, \mu^{(0)})$ sets $\mu_c := \hat{\mu}_a + \hat{\mu}_b + \hat{\mu}_c$ whereas $\mathsf{share}^{u_1}(f, \mu^{(0)})$ chooses it uniformly at random as $\mu_c \xleftarrow{\$} \mathbb{Z}_p$ (or vice-versa in the unpebble case). These distributions are perfectly indistinguishable by the following reason. By the pebbling rule for AND gate, either g_a or g_b must be pebbled. We consider the case g_a is pebbled (we can similarly treat the case g_b is pebbled). When g_a is pebbled, $\hat{\mu}_a$ is independently and uniformly random in both $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$ since the fan-out of g_a is 1 and $\hat{\mu}_a$ does not appear in any other place. Therefore, $\mu_c = \hat{\mu}_a + \hat{\mu}_b + \hat{\mu}_c$ in $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ is independently and uniformly random. The indistinguishability holds even if $\{\mathbf{w}_i\}_{i\in[n]}$ is given to $\mathcal{A}$ since this argument is purely information theoretic. Therefore, these two neighboring hybrids are perfectly indistinguishable in this case.

Pebble/unpebble OR gate with out-going wire c and input wires a, b corresponding to nodes g_a, g_b: The difference between the two games is that $\mathsf{share}^{u_0}(f, \mu^{(0)})$ sets $\mu_{c_a} := \hat{\mu}_a + \hat{\mu}_c$ and $\mu_{c_b} := \hat{\mu}_b + \hat{\mu}_c$ whereas $\mathsf{share}^{u_1}(f, \mu^{(0)})$ chooses them uniformly at random as $\mu_{c_a}, \mu_{c_b} \xleftarrow{\$} \mathbb{Z}_p$ (or vice-versa in the unpebble case). These distributions are perfectly indistinguishable by the following reason. By the pebbling rule for OR gate, both g_a and g_b must be pebbled. When g_a and g_b are pebbled, $\hat{\mu}_a$ and $\hat{\mu}_b$ are independently and uniformly random in both $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$ since the fan-out of g_a and g_b is 1 and $\hat{\mu}_a$ and $\hat{\mu}_b$ do not appear in any other place. Therefore, $\mu_{c_a} = \hat{\mu}_a + \hat{\mu}_c$ and $\mu_{c_b} = \hat{\mu}_b + \hat{\mu}_c$ in $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ are independently and uniformly random. The indistinguishability holds even if $\{\mathbf{w}_i\}_{i\in[n]}$ is given to $\mathcal{A}$ since this argument is purely information theoretic. Therefore, these two neighboring hybrids are perfectly indistinguishable in this case.

In the second and third cases, the indistinguishability between two neighboring games is purely information theoretic and we can simulate all the oracles. In the first case, the indistinguishability between two neighboring games is reduced to the MDDH_k assumption as we see saw in the above. This completes the proof of the following: For all $\ell \in [L]$ and $u_0, u_1 \in \{0,1\}^{R'}$, it holds that

$$\Pr[\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle = 1] - \Pr[\langle \mathcal{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle = 1] \leq n \cdot \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B}}(\kappa).$$

This completes the proof of Lemma 3.4. □

3.4 Security Proof

Here, we prove the following theorem, which addresses the security of the scheme. The proof of the theorem is again similar to the equivalent in [KW19], but with some appropriate adaptations.

Theorem 3.6. *The above construction is adaptively secure under the* MDDH_k *assumption.*

Proof. We prove the theorem by considering a sequence of hybrid games. To define the hybrid distributions, it would be helpful to first give names of various forms of ciphertext and secret keys that will be used. A ciphertext (of message M under attribute x) can be one of the following forms:

Normal: A normal ciphertext is generated as in the scheme.

SF: This is the same as normal ciphertext except that $\mathbf{s}^\top \mathbf{A}$ is replaced by a random vector $\mathbf{c}^\top \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. That is,

$$\mathsf{ct}_x := \left(\mathsf{ct}_1 := \left[\boxed{\mathbf{c}^\top}\right]_1, \mathsf{ct}_2 := \left[\boxed{\mathbf{c}^\top} \sum_{i:x_i=1} \mathbf{W}_i\right]_1, \mathsf{ct}_3 := e\left(\left[\boxed{\mathbf{c}^\top}\right]_1, [\mathbf{k}]_2\right) \cdot M \right).$$

A secret key (for a Boolean formula f) can be one of the following forms:

Normal: A normal key is generated by KeyGen.

SF: An SF key is sampled as a normal key except that $\mathbf{v}$ is replaced by $\mathbf{v}+\delta\mathbf{a}^\perp$, where a fresh δ is chosen per SF key and $\mathbf{a}^\perp$ is any fixed $\mathbf{a}^\perp \in \mathbb{Z}_p^{k+1}\backslash\{\mathbf{0}\}$. That is,

$$\mathsf{sk}_f = \left(\left\{\mathsf{sk}_j := [\mathbf{r}_j]_2, \quad \mathsf{sk}_{\rho(j),j} := [\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2, \quad \{\ \mathsf{sk}_{i,j} := [\mathbf{W}_i\mathbf{r}_j]_2\ \}_{i\in[n]\backslash\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right)$$

where $(\{\mathbf{v}_j\}_{j\in[\hat{m}]},\rho) \xleftarrow{\$} \mathsf{share}(f, \boxed{\mathbf{v}+\delta\mathbf{a}^\perp})$, $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_p^k$.

We then define the following sequence of games to prove the security. Let the number of key generation queries made by an adversary be Q.

- H_0 : This is the real security game for adaptive security where all ciphertexts and keys are normal.
- H_1 : This game is the same as H_0 except that the challenge ciphertext is SF.
- $\mathsf{H}_{2,\ell}$: This game is the same as H_1 except that the first ℓ keys are SF and the remaining $Q-\ell$ keys are normal. The game is defined for $\ell = 0,1,\ldots,Q$.
- H_3 : This is the same as H_Q except that the message to be encrypted is replaced by a random group element $\widetilde{M}$.

Let us fix a PPT adversary $\mathcal{A}$ and denote the advantage of $\mathcal{A}$ in H_{xx} by $\mathsf{Adv}_{\mathrm{xx}}$. We can easily see that $\mathsf{H}_1 \equiv \mathsf{H}_{2,0}$ and $\mathsf{Adv}_3 = 0$. Therefore, to complete the proof of Theorem 3.6, it suffices to prove Lemmata 3.7 to 3.9 in the following. Note that proofs for Lemmata 3.7 and 3.9 are exactly the same as their counterparts in [KW19], whereas we need some adaptations for the proof of Lemma 3.8. For the sake of completeness, we give all the proofs in the following.

Lemma 3.7. *Under the* MDDH_k *assumption on* G_1*, we have* $|\Pr[\langle\mathcal{A},\mathsf{H}_0\rangle=1] - \Pr[\langle\mathcal{A},\mathsf{H}_1\rangle=1]| = \mathsf{negl}(\kappa)$.

Proof. For the sake of contradiction, we assume that $\mathcal{A}$ distinguishes H_0 and H_1 with non-negligible advantage and show that we can construct another adversary $\mathcal{B}$ that solves the MDDH_k assumption. On input $(\mathbb{G},[\mathbf{A}]_1,[\mathbf{z}]_1)$, where either $\mathbf{z}^\top = \mathbf{s}^\top\mathbf{A}$ or $\mathbf{z}=\mathbf{c}$ for $\mathbf{s}\xleftarrow{\$}\mathbb{Z}_p^k$ and $\mathbf{c}\xleftarrow{\$}\mathbb{Z}_p^{k+1}$, $\mathcal{B}$ proceeds as follows.

Setup. $\mathcal{B}$ chooses $\mathbf{W}_i$ for $i\in[n]$ and $\mathbf{v}$ and sets mpk and msk as in the scheme. Note that $[\mathbf{A}\mathbf{W}_i]_1$ can be computed from $[\mathbf{A}]_1$ and $\mathbf{W}_i$.

Secret Keys. $\mathcal{B}$ can answer any secret key query made by $\mathcal{A}$ because it has msk.

Ciphertext. When $\mathbf{A}$ asks for the challenge ciphertext with respect to messages (M_0,M_1) and attribute x, $\mathcal{B}$ samples $\mathsf{coin}\leftarrow\{0,1\}$ and sets the challenge ciphertext as

$$\mathsf{ct}_x := \left([\mathbf{z}^\top]_1, \quad \left[\mathbf{z}^\top\sum_{i:x_i=1}\mathbf{W}_i\right]_1, \quad e([\mathbf{z}^\top]_1,[\mathbf{v}]_2)\cdot M_{\mathsf{coin}}\right).$$

Guess. When $\mathcal{A}$ halts with output coin', $\mathcal{B}$ outputs 1 if $\mathsf{coin}'=\mathsf{coin}$ and 0 otherwise.

It is straightforward to see that $\mathcal{B}$ simulates H_0 for $\mathcal{A}$ when $\mathbf{z}=\mathbf{s}^\top\mathbf{A}$ and H_1 otherwise. This completes the proof of Lemma 3.7. □

Lemma 3.8. *Under the* MDDH_k *assumption on* G_2*, we have* $|\Pr[\langle\mathcal{A},\mathsf{H}_{2,\ell-1}\rangle=1] - \Pr[\langle\mathcal{A},\mathsf{H}_{2,\ell}\rangle=1]| = \mathsf{negl}(\kappa)$ *for* $\ell\in[Q]$.

Proof. For the sake of contradiction, we assume that $\mathcal{A}$ distinguishes $\mathsf{H}_{2,\ell-1}$ and $\mathsf{H}_{2,\ell}$ with non-negligible advantage and show that we can construct another adversary $\mathcal{B}$ that distinguishes the oracles in Lemma 3.2. By the same lemma, this implies an attacker against the MDDH_k assumption. Given $\mu^{(0)}$ as an input and equipped with oracles $\mathcal{O}_{\mathsf{F},\beta}$, $\mathcal{O}_{\mathsf{X}}$, and $\mathcal{O}_{\mathsf{E}}$, $\mathcal{B}$ proceeds as follows.

Setup. First, $\mathcal{B}$ chooses $\mathbf{A}\xleftarrow{\$}\mathbb{Z}_p^{k\times(k+1)}$, $\widetilde{\mathbf{W}}_i\xleftarrow{\$}\mathbb{Z}_p^{(k+1)\times k}$ for $i\in[n]$, and $\tilde{\mathbf{v}}\xleftarrow{\$}\mathbb{Z}_p^{k+1}$, computes $\mathbf{a}^\perp\in\mathbb{Z}_p^{k+1}\backslash\{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{a}^\perp=\mathbf{0}$. It then sets $\widetilde{\mathbf{W}}_0 := \mathbf{0}$ and implicitly defines

$$\mathbf{v} := \tilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^\perp, \quad \mathbf{W}_i := \widetilde{\mathbf{W}}_i + \mathbf{a}^\perp\mathbf{w}_i^\top \quad \text{for} \quad i\in[0,n]$$

where $\mathbf{w}_i \in \mathbb{Z}_p^k$ and $\mu^{(0)} \in \mathbb{Z}_p$ are chosen by the game. Then, $\mathcal{B}$ gives

$$\mathsf{mpk} := \Big([\mathbf{A}]_1, [\mathbf{A}\widetilde{\mathbf{W}}_1]_1, \ldots, [\mathbf{A}\widetilde{\mathbf{W}}_n]_1, e([\mathbf{A}]_1, [\mathbf{v}]_2)\Big)$$

to $\mathcal{A}$.

Secret Keys. B handles the secret key queries made by $\mathcal{A}$ as follows.

- For the first $\ell - 1$ secret key queries, say for formula f of size m, $\mathcal{B}$ computes

$$(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \underbrace{\tilde{\mathbf{v}} + \tilde{\delta}\mathbf{a}^\perp}_{=\mathbf{v}+\delta\mathbf{a}^\perp})$$

  where $\tilde{\delta} \xleftarrow{\$} \mathbb{Z}_p$ is drawn independently for each key (here, the per-key $\delta = \tilde{\delta} - \mu^{(0)}$ implicitly). Next, for each $j \in [\hat{m}]$, it queries $\mathcal{O}_{\mathsf{E}} \to ([\mathbf{r}_j]_2, \{[\mathbf{w}_i^\top \mathbf{r}_j]_2\}_{i\in[n]})$ and forms the following SF key as

$$\mathsf{sk}_f = \left(\left\{[\mathbf{r}_j]_2, \quad \underbrace{[\mathbf{v}_j + \widetilde{\mathbf{W}}_{\rho(j)}\mathbf{r}_j + \mathbf{a}^\perp \mathbf{w}_{\rho(j)}^\top \mathbf{r}_j]_2}_{=\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j}, \quad \left\{\underbrace{[\widetilde{\mathbf{W}}_i \mathbf{r}_j + \mathbf{a}^\perp \mathbf{w}_i^\top \mathbf{r}_j]_2}_{=\mathbf{W}_i \mathbf{r}_j}\right\}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right).$$

  Then, it returns sk_f to $\mathcal{A}$.

- For the last $Q - \ell$ secret key queries, say for formula f of size m, $\mathcal{B}$ proceeds as before for the first $\ell - 1$ queries except

$$(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \underbrace{\tilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^\perp}_{=\mathbf{v}}).$$

  It is easy to see that it forms a normal key.

- For the ℓ-th secret key queries, say for formula f of size m, $\mathcal{B}$ computes $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \tilde{\mathbf{v}})$, queries $\mathcal{O}_{\mathsf{F},\beta}(f) \to \left(\left\{[\mathbf{r}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}\mathbf{r}_j]_2, \left\{[\mathbf{w}_i^\top \mathbf{r}_j]_2\right\}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right)$ and uses these components to return:

$$\mathsf{sk}_f = \left(\left\{[\mathbf{r}_j]_2, \quad \underbrace{[\mathbf{v}_j + \widetilde{\mathbf{W}}_{\rho(j)}\mathbf{r}_j + \mathbf{a}^\perp(\mu_j + \mathbf{w}_{\rho(j)}^\top \mathbf{r}_j)]_2}_{=(\mathbf{v}_j + \mu_j\mathbf{a}^\perp) + \mathbf{W}_{\rho(j)}\mathbf{r}_j}, \quad \left\{\underbrace{[\widetilde{\mathbf{W}}_i \mathbf{r}_j + \mathbf{a}^\perp \mathbf{w}_i^\top \mathbf{r}_j]_2}_{=\mathbf{W}_i \mathbf{r}_j}\right\}_{i\in[n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right)$$

  We claim that if $\beta = 0$, then sk_f is a normal key, and if $\beta = 1$, then sk_f is an SF key. This follows from the fact that thanks to linearity, the shares

$$(\{\mathbf{v}_j + \mu_j\mathbf{a}^\perp\}_{j\in[\hat{m}]}, \rho), \quad \text{where} \quad (\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \tilde{\mathbf{v}}), \quad (\{\mu_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \mu^{(\beta)})$$

  are identically distributed to $\mathsf{share}(f, \tilde{\mathbf{v}} + \mu^{(\beta)}\mathbf{a}^\perp)$. The claim then follows from the fact that $\mathbf{v} = \tilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^\perp$ and that $\tilde{\mathbf{v}} + \mu^{(1)}\mathbf{a}^\perp$ is identically distributed to $\mathbf{v} + \delta\mathbf{a}^\perp$, where we set $\delta := \mu^{(1)} - \mu^{(0)}$ is a fresh random value for the key.

Ciphertext. When $\mathbf{A}$ asks for the challenge ciphertext with respect to messages (M_0, M_1) and attribute x, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{X}}$ on input x to obtain $\{\mathbf{w}_i\}_{i:x_i=1}$ It then samples $\mathsf{coin} \leftarrow \{0,1\}$, $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and returns the following challenge ciphertext for $\mathcal{A}$:

$$\mathsf{ct}_x := \left([\mathbf{c}^\top]_1, \quad \left[\mathbf{c}^\top \sum_{i:x_i=1} (\underbrace{\widetilde{\mathbf{W}}_i + \mathbf{a}^\perp \mathbf{w}_i^\top}_{=\mathbf{W}_i})\right]_1, \quad e([\mathbf{c}^\top]_1, [\tilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^\perp]_2) \cdot M_{\mathsf{coin}}\right).$$

Guess. When $\mathcal{A}$ halts with output coin', $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$ and 0 otherwise.

Putting everything together, we can see that $\mathcal{B}$ simulates $\mathsf{H}_{2\ell-1}$ for $\mathcal{A}$ when $\beta = 0$ and $\mathsf{H}_{2\ell}$ otherwise. This completes the proof of Lemma 3.8. □

Lemma 3.9. *We have* $|\Pr[\langle \mathcal{A}, \mathsf{H}_{2,\ell}\rangle = 1] - \Pr[\langle \mathcal{A}, \mathsf{H}_3\rangle = 1]| = 1/p$ *unconditionally.*

Proof. We show that these two hybrids are identically distributed conditioned on $\mathbf{c}^\top \mathbf{a}^\perp \neq 0$. To see this, consider two ways of sampling $\mathbf{v}$: as $\tilde{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and as $\tilde{\mathbf{v}} + \tilde{m}\mathbf{a}^\perp$ for an independent $\tilde{m} \xleftarrow{\$} \mathbb{Z}_p$. Note that both result in $\mathbf{v}$ having a uniform distribution.

Using $\tilde{\mathbf{v}}$ to simulate hybrid $\mathsf{H}_{2,Q}$ obviously results in $\mathsf{H}_{2,Q}$ (where $\mathbf{v} = \tilde{\mathbf{v}}$). However, using the identically distributed $\mathbf{v} = \tilde{\mathbf{v}} + \tilde{m}\mathbf{a}^\perp$ to simulate $\mathsf{H}_{2,Q}$ results in H_3 with $\widetilde{M} = M \cdot e([\mathbf{c}^\top]_1, [\tilde{m}\mathbf{a}^\perp]_2)$ and re-defined randomness $\tilde{\delta}_j := \delta_j + \tilde{m}$ in the secret keys. Note that the information of $\tilde{m}$ is not revealed to $\mathcal{A}$ from the secret key queries because $\tilde{m}$ is masked by per-key randomness δ_j. Therefore, $\widetilde{M}$ is distributed uniformly at random over G_T as long as $\mathbf{c}^\top \mathbf{a} \neq 0$.

Since $\mathbf{c}$ is chosen at random and independent from $\mathbf{a}^\perp \neq \mathbf{0}$, so $\mathbf{c}^\top \mathbf{a}^\perp = 0$ with probability $1/p$, and since we know that $\mathsf{H}_{2,Q} \equiv \mathsf{H}_3$ conditioned on $\mathbf{c}^\top \mathbf{a}^\perp \neq 0$, the lemma follows. □

This completes the proof of Theorem 3.6. □

4 Compact Constrained Signature

4.1 Constrained Signature

We provide definition of a constrained signature (CS) scheme. We also provide an additional feature (i.e., online/offline efficiency) for CS schemes which will play a vital role in our compact NIZK construction in Section 5.

Definition 4.1 (Constrained Signature). *A constrained signature (CS) scheme with message space* $\{0,1\}^n$ *for a circuit class* $\mathcal{C} = \{C : \{0,1\}^n \to \{0,1\}\}$ *consists of PPT algorithms* $(\mathsf{CS.Setup}, \mathsf{CS.KeyGen}, \mathsf{CS.Sign}, \mathsf{CS.Vrfy})$.

$\mathsf{CS.Setup}(1^\kappa, 1^n) \to (\mathsf{msk}, \mathsf{vk})$*: The setup algorithm on input the security parameter* 1^λ *and the input length* 1^n*, outputs a master secret key* msk *and a verification key* vk.

$\mathsf{CS.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$*: The key generation algorithm on input a master secret key* msk *and a circuit* $C \in \mathcal{C}$*, outputs a signing key* sk_C.

$\mathsf{CS.Sign}(\mathsf{sk}_C, x) \to \sigma$*: The signing algorithm on input the signing key* sk_C *and message* $x \in \{0,1\}^n$*, outputs a signature* σ.

$\mathsf{CS.Vrfy}(\mathsf{vk}, x, \sigma) \to \top$ *or* $\perp$*: The verification algorithm on input the verification key* vk*, message* x*, and signature* σ*, outputs either* $\perp$ *(indicating the signature is valid) or* $\top$ *(indicating the signature is invalid).*

A CS scheme must satisfy the following requirements.

Correctness. For all $\kappa \in \mathbb{N}$, $n = n(\kappa) \in \mathbb{N}$, $(\mathsf{msk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^n)$, $x \in \{0,1\}^n$, $C \in \mathcal{C}$ such that $C(x) = 1$, and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$, we have

$$\Pr[\mathsf{CS.Vrfy}(\mathsf{vk}, x, \mathsf{CS.Sign}(\mathsf{sk}_C, x)) = \top] = 1$$

Unforgeability. We define (adaptive) unforgeability for a CS scheme. The security notion is defined by the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger runs $(\mathsf{msk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^n)$ and gives vk to $\mathcal{A}$. It also prepares an empty list $\mathcal{Q}$.

Key Queries: $\mathcal{A}$ can adaptively make key queries unbounded polynomially many times throughout the game. When $\mathcal{A}$ queries $C \in \mathcal{C}$, the challenger runs $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$ and returns sk_C to $\mathcal{A}$. Finally, the challenger updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{C\}$.

Forgery: Eventually, $\mathcal{A}$ outputs (x^*, σ^*) as the forgery. We say $\mathcal{A}$ *wins* if $\mathsf{CS.Vrfy}(\mathsf{vk}, x^*, \sigma^*) = \top$ holds. Furthermore, we say that $\mathcal{A}$ is *admissible* if $C(x^*) = 0$ holds for all $C \in \mathcal{Q}$ at the end of the game.

We say the CS scheme is (adaptively) *unforgeable* if the winning probability for all admissible PPT adversaries $\mathcal{A}$ in the above game is $\mathsf{negl}(\kappa)$, where the probability is taken over the randomness of all algorithms.

The following property is optional in the sense that our CS scheme can achieve the following property, but the property is not strictly necessary for our application of CS to the construction of compact NIZKs.

Context-Hiding (optional). For all $\kappa, n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^n)$, $x \in \{0,1\}^n$, $C_0, C_1 \in \mathcal{C}$, $(\mathsf{msk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^n)$, $\mathsf{sk}_{C_0} \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C_0)$, and $\mathsf{sk}_{C_1} \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C_1)$, we need that the following distributions are statistically close:

$$\{\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_{C_0}, x)\} \overset{\text{stat}}{\approx} \{\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_{C_1}, x)\}$$

where the probability is only over the randomness used by $\mathsf{CS.Sign}$.

Additionally to the above essential requirements for CS, we introduce a natural notion of *decomposable online-offline efficiency*. At a high level, this notion states that if we (partially) knew the message x to be signed in advance, then we can modify the verification key vk to a message specific verification key vk_x which allows for an efficient verification of signature σ with running time independent of $|x|$. More formally, the notion is defined as follows.

Definition 4.2 (Decomposable Online-Offline Efficiency). *A constrained signature with message space $\{0,1\}^n$ for a circuit class $\mathcal{C} = \{C : \{0,1\}^n \to \{0,1\}\}$ is said to have decomposable online-offline efficiency if there further exists PPT algorithms $(\mathsf{CS.Aggrgt}, \mathsf{CS.VrfyOnL})$ exhibiting the following properties.*

- *The verification key vk can be decomposed into $\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b} \in \mathcal{VK}\}_{i\in[n],b\in\{0,1\}})$, where $\mathcal{VK}$ is a space of verification key component.*
- *Any component in $\mathcal{VK}$, any honestly generated vk_0, and any honestly generated signature σ can be represented as binary strings of fixed polynomial length $\mathsf{poly}(\kappa)$. In particular, length of these components are independent from n.*
- *Algorithm $\mathsf{CS.Aggrgt}$ takes as input an element of $\mathcal{VK}^* = \cup_{\ell\in\mathbb{N}}\mathcal{VK}^\ell$ and outputs an element in $\mathcal{VK}$. We require that for any $y, z \in \{0,1\}^*$ such that $x = y\|z \in \{0,1\}^n$, we have*

$$\begin{aligned}&\mathsf{CS.Aggrgt}\left(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]}\right)\\ =&\mathsf{CS.Aggrgt}\left(\mathsf{CS.Aggrgt}\left(\{\mathsf{vk}_{i,y_i}\}_{i\in[|y|]}\right), \mathsf{CS.Aggrgt}\left(\{\mathsf{vk}_{|y|+i,z_i}\}_{i\in[|z|]}\right)\right).\end{aligned}$$

- *Algorithm $\mathsf{CS.VrfyOnL}$ takes as input vk_0, a component in $\mathcal{VK}$ and a signature in σ, and outputs either $\top$ or $\bot$. We require that for any $x \in \{0,1\}^n$, for any honestly generated vk, and for any (possibly maliciously generated) σ, we have*

$$\mathsf{CS.Vrfy}(\mathsf{vk}, x, \sigma) = \mathsf{CS.VrfyOnL}\left(\mathsf{vk}_0, \mathsf{CS.Aggrgt}\left(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]}\right), \sigma\right).$$

  Observe that the input length of $\mathsf{CS.VrfyOnL}$ is independent from n, which follows from the second item of this definition. We require that the running time of $\mathsf{CS.VrfyOnL}$ is independent from n as well.

4.2 Construction and Security

Here, we give the construction of our constrained signature (CS) scheme that will be used for the construction of the compact NIZK. The CS scheme has very compact signature size and the decomposable online-offline efficiency defined in Definition 4.2. In order to get the CS scheme, we apply the folklore conversion that converts ABE into CS to our compact KP-ABE scheme in Section 3, where the signing key sk_f for the function f in the CS scheme is the same as the secret key sk_f for the same function f in the ABE scheme, and the signature on a string x in the CS scheme is certain "aggregated form" of the secret key that is derived when decrypting an ABE ciphertext encrypted for the attribute x. To verify a signature on x in the CS, we encrypt a random message for x in the underlying ABE and then see if the message is recovered or not when decrypting the ciphertext using the signature as an (aggregated form of) secret key.

The CS scheme obtained by the above conversion can only deal with monotone Boolean formulae, since the original ABE is for the same class of functions. For our purpose, we need CS scheme for $\mathbf{NC}^1$ circuits, which is more general class than monotone Boolean formulae. This gap can be filled using Lemma 2.9.

We then provide the description of the construction.

$\mathsf{CS.Setup}(1^\kappa, 1^n)$**:** Run $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}$, $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times k}$ for $i \in [2n]$ and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and output

$$\mathsf{vk} = ([\mathbf{A}]_1, [\mathbf{A}\mathbf{W}_1]_1, \ldots, [\mathbf{A}\mathbf{W}_{2n}]_1, e([\mathbf{A}]_1, [\mathbf{v}]_2)), \quad \mathsf{msk} = (\mathbf{v}, \mathbf{W}_1, \ldots, \mathbf{W}_{2n}).$$

$\mathsf{CS.KeyGen}(\mathsf{msk}, C)$**:** To generate a signing key for a circuit C, run $\mathsf{EncCir}(C) \to f$. Then sample $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \mathbf{v})$ and $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [\hat{m}]$ and output sk_f, which consists of the following.

$$\left(\left\{\mathsf{sk}_j := [\mathbf{r}_j]_2,\ \mathsf{sk}_{\rho(j),j} := [\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2,\ \{\mathsf{sk}_{i,j} := [\mathbf{W}_i\mathbf{r}_j]_2\ \}_{i\in[2n]\setminus\{\rho(j)\}}\right\}_{j\in[\hat{m}]}\right)$$

where $\mathbf{W}_0 = \mathbf{0}$ and $\hat{m}$ is the number of shares that are generated by $\mathsf{share}(f, \mathbf{v})$.

$\mathsf{CS.Sign}(\mathsf{sk}_f, x)$**:** Set $\hat{x} := \mathsf{EncInp}(x)$ and compute ω_j such that $\mathbf{v} = \sum_{j:\rho(j)=0\vee\hat{x}_{\rho(j)}=1} \omega_j \mathbf{v}_j$ and output

$$\sigma = \left(\sigma_1 = \prod_{j:\rho(j)=0\vee\hat{x}_{\rho(j)}=1} \left(\prod_{i:\hat{x}_i=1} \mathsf{sk}_{i,j}\right)^{\omega_j},\quad \sigma_2 = \prod_{j:\rho(j)=0\vee\hat{x}_{\rho(j)}=1} \mathsf{sk}_j^{\omega_j}\right).$$

$\mathsf{CS.Vrfy}(\mathsf{vk}, x, \sigma)$**:** Parse $\sigma \to (\sigma_1, \sigma_2) \in G_2^k \times G_2^k$ and output $\perp$ if the signature is not in this form. Otherwise, compute $\hat{x} = \mathsf{EncInp}(x)$ and

$$\mathsf{vk}' = \prod_{i:\hat{x}_i=1} [\mathbf{A}\mathbf{W}_i]_1. \tag{5}$$

Then output $\top$ if the following holds and $\perp$ otherwise:

$$e([\mathbf{A}]_1, \sigma_1) \cdot e(\mathsf{vk}', \sigma_2)^{-1} = e([\mathbf{A}]_1, [\mathbf{v}]_2).$$

Correctness. The correctness follows since we have $f(\hat{x}) = 1$ when $C(x) = 1$ from Lemma 2.9 and

$$\sigma_1 = \left[\mathbf{v} + \sum_{i:\hat{x}_i=1} \mathbf{W}_i\mathbf{r}\right]_2,\quad \sigma_2 = [\mathbf{r}]_2,\quad \text{where}\quad \mathbf{r} = \sum_{j:\rho(j)=0\vee\hat{x}_{\rho(j)}=1} \omega_j\mathbf{r}_j. \tag{6}$$

Online-Offline Decomposability.

Theorem 4.3. *The CS scheme above has decomposable online-offline efficiency defined as per Definition 4.2.*

Proof. To prove the theorem, we define $\mathcal{VK}$, vk_0, and $\mathsf{vk}_{i,b}$ for $i \in [n]$, $b \in \{0, 1\}$ as

$$\mathcal{VK} := G_1^{k\times k},\quad \mathsf{vk}_0 := ([\mathbf{A}]_1, e([\mathbf{A}]_1, [\mathbf{v}]_2)),\quad \mathsf{vk}_{i,b} := [\mathbf{A}\mathbf{W}_{2i-b}]_1.$$

It is easy to see that the first and the second items in Definition 4.2 are satisfied. We then define additional algorithms $\mathsf{CS.VrfyOnL}$ and $\mathsf{CS.Aggrgt}$ as follows:

$\mathsf{CS.Aggrgt}(\{\mathsf{vk}_i\}_{i\in[n']})$**:** If there exists $i \in [n']$ such that $\mathsf{vk}_i \notin \mathcal{VK} = G_1^{k\times k}$, output $\perp$. Otherwise, output

$$X := \prod_{i\in[n']} \mathsf{vk}_i,$$

where the product represents the component-wise multiplication in G_1.

$\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, \mathsf{vk}', \sigma)$**:** Parse $\mathsf{vk}_0 \to (A \in G_1^{k\times(k+1)}, V \in G_T^k)$, $\mathsf{vk}' \in G_1^{k\times k}$, and $\sigma \to (\sigma_1, \sigma_2) \in G_2^k \times G_2^k$. Then output $\top$ if the following holds and $\bot$ otherwise:

$$e(A, \sigma_1) \cdot e(\mathsf{vk}', \sigma_2)^{-1} = V.$$

The third item in Definition 4.2 follows from the fact that the following equation holds for any $x = y\|z \in \{0,1\}^n$:

$$\begin{aligned}
\prod_{i\in[2n]} \underbrace{[\mathbf{AW}_{i,2i-x_i}]}_{=\mathsf{vk}_{i,x_i}} &= \prod_{i\in[|y|]} [\mathbf{AW}_{i,2i-x_i}] \cdot \prod_{i\in[|y|+1,|y|+|z|]} [\mathbf{AW}_{i,2i-x_i}] \\
&= \prod_{i\in[|y|]} [\mathbf{AW}_{i,2i-y_i}] \cdot \prod_{i\in[|y|+1,|y|+|z|]} [\mathbf{AW}_{i,2i-z_{i-|y|}}] \\
&= \prod_{i\in[|y|]} \underbrace{[\mathbf{AW}_{i,2i-y_i}]}_{=\mathsf{vk}_{i,y_i}} \cdot \prod_{j\in[|z|]} \underbrace{[\mathbf{AW}_{|y|+j,2(|y|+j)-z_j}]}_{=\mathsf{vk}_{|y|+j,z_j}} \cdot
\end{aligned}$$

To prove the fourth item, it suffices to show that vk' computed as Equation (5) equals to $\mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$. This follows since the former is the product of $[\mathbf{AW}_i]_1$ over i in $S := \{i \in [2n] : \hat{x}_i = 1\}$ and the latter is over i in $S' := \{2j - x_j : j \in [n]\}$, and we have $S = S'$ by the definition of $\hat{x}$ (See Lemma 2.9). □

Security. In the following, we show that the above construction is unforgeable and then discuss how to extend the scheme to satisfy context-hiding. While the latter property is not necessary for our application of CS in Section 5, this property may be useful when we use the CS scheme stand-alone.

Theorem 4.4. *The above construction is (adaptively) unforgeable under the* MDDH_k *assumption.*

Proof. For the sake of contradiction, suppose that there exists an adversary $\mathcal{A}$ that breaks unforgeability of the Π_{CS} with non-negligible probability ϵ. We then construct a PPT adversary $\mathcal{B}$ that breaks the adaptive security of the ABE with advantage ϵ for the attribute length $2n$ as follows.

$\mathcal{B}(\mathsf{mpk})$**:** It sets $\mathsf{vk} := \mathsf{mpk}$ and gives the master public key to $\mathcal{A}$. When $\mathcal{A}$ makes a signing key query for a circuit C, $\mathcal{B}$ runs $\mathsf{EncCir}(C) \to f$ and makes a key generation query for f to obtain sk_f. Then, $\mathcal{B}$ passes sk_f to $\mathcal{A}$. At some point, $\mathcal{A}$ outputs a forgery (x^*, σ^*). Then, $\mathcal{B}$ outputs a random bit and abort if $\mathsf{CS.Vrfy}(\mathsf{vk}, x^*, \sigma^*) = \bot$. Otherwise, $\mathcal{B}$ samples two random distinctive messages $M_0, M_1 \in G_T$ and makes a challenge query for $(\hat{x}^*, (M_0, M_1))$, where $\hat{x}^* = \mathsf{EncInp}(x^*)$. Given the challenge ciphertext ct, it first parses $\mathsf{ct} \to (\mathsf{ct}_1 \in G_1^{k+1}, \mathsf{ct}_2 \in G_1^k, \mathsf{ct}_3 \in G_T)$ and $\sigma^* \to (\sigma_1^* \in G_2^{k+1}, \sigma_2^* \in G_2^{k+1})$ and computes $M' := e(\mathsf{ct}_1, \sigma_1^*)^{-1} \cdot e(\mathsf{ct}_2, \sigma_2^*) \cdot \mathsf{ct}_3$. It outputs 0 if $M' = M_0$ and 1 otherwise.

We first check that $\mathcal{B}$ is an admissible adversary if so is $\mathcal{A}$, since we have $C(x^*) = 0$ iff $f(\hat{x}^*) = 0$ for any C and $f = \mathsf{EncCir}(C)$ from Lemma 2.9. We then claim that whenever $\mathsf{CS.Vrfy}(\mathsf{vk}, x^*, \sigma^*) = \top$, we have $M' = M_{\mathsf{coin}}$. To prove the claim, let us assume that $\mathsf{CS.Vrfy}(\mathsf{vk}, \hat{x}^*, \sigma^*) = \top$ holds. Then, we have

$$e([\mathbf{A}]_1, \sigma_1^*) \cdot e(\prod_{i:\hat{x}_i^*=1} [\mathbf{AW}_i]_1, \sigma_2^*)^{-1} = e([\mathbf{A}]_1, [\mathbf{v}]_2)$$

by the definition of $\mathsf{CS.Vrfy}$. Furthermore, there exists $\mathbf{s} \in \mathbb{Z}_p^k$ such that $\mathsf{ct}_1 = [\mathbf{s}^\top \mathbf{A}]_1$, $\mathsf{ct}_2 = [\mathbf{s}^\top \sum_{i:y_i^*=1} \mathbf{AW}_i]_1$, and $\mathsf{ct}_3 = e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2) \cdot M_{\mathsf{coin}}$ by the definition of Enc. Then, the above equation implies

$$e(\mathsf{ct}_1, \sigma_1^*) \cdot e(\mathsf{ct}_2, \sigma_2^*)^{-1} = e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2)$$

which in turns implies $M' = M_{\mathsf{coin}}$. Thus, $\mathcal{B}$ correctly guesses coin when $\mathcal{A}$ breaks the unforgeability of Π_{CS} and outputs a random bit otherwise. This implies that the advantage of $\mathcal{B}$ is ϵ, which is non-negligible as desired. □

Remark 4.5 (Adding Context-Hiding for the Scheme). We remark that it is possible to make the above scheme context-hiding by adding the following modification. Namely, we change the scheme so that it contains

$$[\mathbf{R}]_2, [\mathbf{W}_1\mathbf{R}]_2, \ldots, [\mathbf{W}_{2n}\mathbf{R}]_2,$$

for random $\mathbf{R} \in \mathbb{Z}_p^{k\times k}$ in vk. This modification allows us to randomize $\mathbf{r}$ in Equation (6), which makes the scheme context-hiding. We discuss that even with this change, the scheme remains adaptively unforgeable. For proving this, it suffices to show that our KP-ABE scheme in Section 3 remains adaptively secure even if we add $([\mathbf{R}]_2, [\mathbf{W}_1\mathbf{R}]_2, \ldots, [\mathbf{W}_n\mathbf{R}]_2)$ to the master public key. To show this, we explain the necessary modifications for the proof of Theorem 3.6, which consists of Lemmata 3.7 to 3.9. We first observe that the proof of Lemma 3.7 is unchanged by this change, since the reduction algorithm in the proof knows all of $\{\mathbf{W}_i\}_i$ and can simulate the extra terms. Similarly, the proof for Lemma 3.9 is unchanged, since this is shown by an information theoretic argument that holds even if the adversary knows all of $\{\mathbf{W}_i\}_i$. As for the proof of Lemma 3.8, slight care is needed since $\{\mathbf{W}_i\}_i$ are not known to the simulator. In this case, the simulator makes multiple queries for $\mathcal{O}_{\mathsf{E}}()$ to get $\{[\mathbf{r}_j]_2, \{[\mathbf{w}_i^\top \mathbf{r}]_2\}_{i\in[n]}\}_{j\in[k]}$ and forms $([\mathbf{R}]_2, \{[\mathbf{w}_i^\top \mathbf{R}]_2\}_{i\in[n]})$, where $\mathbf{R} \in \mathbb{Z}_p^{k\times k}$ is the concatenation of $\{\mathbf{r}_j\}_j$. Then the simulator can simulate the terms $\{[\mathbf{W}_i\mathbf{R}]_2\}_{i\in[n]}$ from $([\mathbf{R}]_2, \{[\mathbf{w}_i^\top \mathbf{R}]_2\}_{i\in[n]})$, $\mathbf{a}^\perp$, and $\widetilde{\mathbf{W}}_i$, since we have

$$[\mathbf{W}_i\mathbf{R}]_2 = [\widetilde{\mathbf{W}}_i\mathbf{R} + \mathbf{a}^\perp(\mathbf{w}_i^\top \mathbf{R})]_2.$$

5 Compact NIZK from Compact Constrained Signatures

5.1 Main Construction

Here, we construct a compact NIZK based on the compact CS scheme which we constructed in Section 4. 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 $(x,w) \in \mathcal{R}$.
The construction will be given by combining following ingredients.

- A symmetric key encryption (SKE) scheme $\Pi_{\mathsf{SKE}} = (\mathsf{SKE.Setup}, \mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with message space $\{0,1\}^m$, key space $\{0,1\}^\ell$ and ciphertext space $\{0,1\}^{|\mathsf{ct}|}$. We require that it is one-time secure, its decryption circuit can be computed in $\mathbf{NC}^1$, it has an additive ciphertext overhead (i.e., $|\mathsf{ct}| = m + \mathsf{poly}(\kappa)$), and ℓ is independent of m. As shown in Lemma 2.3, such an SKE scheme exists under the CDH assumption in a subgroup of $\mathbb{Z}_p^*$.
- A constrained signature scheme $(\mathsf{CS.Setup}, \mathsf{CS.KeyGen}, \mathsf{CS.Sign}, \mathsf{CS.Vrfy}, \mathsf{CS.Aggrgt}, \mathsf{CS.VrfyOnL})$ we constructed in Section 4. The scheme should support the circuit $f_{\mathsf{pp}_{\mathsf{SKE}}}$ that computes

  $$f_{\mathsf{pp}_{\mathsf{SKE}}}(K, x, \mathsf{ct}) = C(x, \mathsf{SKE.Dec}(\mathsf{pp}_{\mathsf{SKE}}, K, \mathsf{ct}))$$

  for all $\mathsf{pp}_{\mathsf{SKE}} \in \mathsf{SKE.Setup}(1^\kappa)$.
- (Not necessarily compact) extractable NIZK scheme $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for the language corresponding to the relation $\widetilde{\mathcal{R}}$ defined below:
  $((\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), (K, \sigma)) \in \widetilde{\mathcal{R}}$ if and only if the followings are satisfied:
  1. $K \in \{0,1\}^\ell$,
  2. $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top$ where $Z = \mathsf{CS.Aggrgt}\big(\mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,K_i}\}_{i\in[\ell]}), Y\big)$

Our compact NIZK is described as follows.

$\mathsf{Setup}'(1^\kappa)$**:**

1. Generate $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$.

2. Generate $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$.
3. Generate $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell+n+|\mathsf{ct}|],b\in\{0,1\}}), \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^{\ell+n+|\mathsf{ct}|})$.
4. Generate $\mathsf{sk}_f \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, f_{\mathsf{pp}_{\mathsf{SKE}}})$.
5. Output $\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk}, \mathsf{sk}_f)$.

$\mathsf{Prove}'(\mathsf{crs}', x, w)$**:**

1. Abort if $\mathcal{R}(x, w) = 0$. Otherwise, do the following.
2. Parse $\mathsf{crs}' \to (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell+n+|\mathsf{ct}|],b\in\{0,1\}}), \mathsf{sk}_f)$.
3. Generate $K \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$ and $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, K, w)$.
4. Compute $\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_f, (K, x, \mathsf{ct}))$.
5. Compute $Y := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i}\}_{i\in[n+|\mathsf{ct}|]})$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$.
6. Compute $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), (K, \sigma))$.
7. Output $\pi' := (\mathsf{ct}, \pi)$.

$\mathsf{Verify}'(\mathsf{crs}', x, \pi')$**:**

1. Parse $\pi' \to (\mathsf{ct}, \pi)$. If it is not in this form, reject it. Otherwise, do the following.
2. Parse $\mathsf{crs}' \to (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell+n+|\mathsf{ct}|],b\in\{0,1\}}), \mathsf{sk}_f)$.
3. Compute $Y := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i}\}_{i\in[n+|\mathsf{ct}|]})$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$.
4. Output $\top$ if $\mathsf{Verify}(\mathsf{crs}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), \pi) = \top$ and otherwise $\bot$.

Correctness. Suppose that (ct, π) is an honestly generated proof on $(x, w) \in \mathcal{R}$. Then we have $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, K, w)$ and $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), (K, \sigma))$ where $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$, $K \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$, $\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_f, (K, x, \mathsf{ct}))$, and

$$Y = \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i}\}_{i\in[n+|\mathsf{ct}|]}).$$

By the correctness of Π_{SKE}, we have $f_{\mathsf{pp}_{\mathsf{SKE}}}(K, x, \mathsf{ct}) = 1$. Furthermore, by the correctness of Π_{CS}, we have $\mathsf{CS.Vrfy}(\mathsf{vk}, (K, x, \mathsf{ct}), \sigma) = \top$, which is equivalent to

$$\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top \quad \text{where} \quad Z = \mathsf{CS.Aggrgt}\big(\mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,K_i}\}_{i\in[\ell]}), Y\big).$$

Therefore we have $((\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), (K, \sigma)) \in \widetilde{\mathcal{R}}$ and thus we have $\mathsf{Verify}(\mathsf{crs}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), \pi) = \top$ by the correctness of Π_{NIZK}.

Efficiency. We first observe that the size of the verification circuit for the relation $\widetilde{\mathcal{R}}$ is $\mathsf{poly}(\kappa)$, which is independent of the size of the verification circuit for $\mathcal{R}$. This is because $Z = \mathsf{CS.Aggrgt}\big(\mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,K_i}\}_{i\in[\ell]}), Y\big)$ can be computed in polynomial time in κ and the length $\ell = \mathsf{poly}(\kappa)$ of K and the running time of $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma)$ does not depend on the length of (x, ct) (and in particular the complexity of the circuit f) as required in Definition 4.2. Therefore, the size of π is $\mathsf{poly}(\kappa)$ and independent of $|x|$, $|w|$, or $|C|$ even though we do not require any compactness requirement for the underlying NIZK Π_{NIZK}. Since we assume $|\mathsf{ct}| = m + \mathsf{poly}(\kappa)$, the total proof size is $|w| + \mathsf{poly}(\kappa)$. We note that this scheme can be directly implemented only when the relation $\mathcal{R}$ can be verified in $\mathbf{NC}^1$. Otherwise, we have to first expand the witness to make the relation verifiable in $\mathbf{NC}^1$ similarly to [GGH+16, KNYY19b]. This is done by considering all values corresponding to all gates when computing the circuit C on input (x, w) to be the new witness and have the new circuit verify the consistency of the values for all gates in C. In this case, the proof size becomes $|C| + \mathsf{poly}(\kappa)$.

Since the relation $\widetilde{R}$ is well-suited to be proven by the Groth-Sahai proof, a fairly efficient instantiation is possible based on the Groth-Sahai proof. Especially, a proof consists of $|C|$ bits, $6\ell + 14$ elements of $\mathbb{G}_1$ and $6\ell + 24$ elements of

$\mathbb{G}_2$ when instantiated under the SXDH assumption where ℓ denotes the key length of the SKE scheme. See Appendix B for more details. We also note that if the relation $\mathcal{R}$ can be verified by a "leveled circuit" [BGI16], we can further reduce the proof size to $|w| + |C|/\log\kappa + \mathsf{poly}(\kappa)$ which is sublinear in $|C|$ similarly to [KNYY19b]. (See [KNYY19b] for details.)

Security. In the following, we prove the soundness and the zero-knowledge property of Π'_{NIZK}.

Theorem 5.1 (Soundness). *The above NIZK scheme Π'_{NIZK} is computationally (adaptive) sound if Π_{NIZK} satisfies extractability and Π_{CS} is unforgeable.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ that breaks soundness. Then we construct a PPT adversary $\mathcal{B}$ that breaks the unforgeability of Π_{CS} as follows.

$\mathcal{B}(\mathsf{vk})$**:** It generates $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$ and queries $f_{\mathsf{pp}_{\mathsf{SKE}}}$ to the key generation oracle to obtain sk_f where $f_{\mathsf{pp}_{\mathsf{SKE}}}$ is the circuit as defined in the description of the scheme. Then it generates $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa; r_{\mathsf{Setup}})$, runs $\mathcal{A}(\mathsf{crs}')$ to obtain $(x^*, \pi'^* = (\mathsf{ct}, \pi))$ where $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk}, \mathsf{sk}_f)$. Then it computes $(K, \sigma) \xleftarrow{\$} \mathsf{Extract}(r_{\mathsf{Setup}}, \pi)$ and outputs $((K, x^*, \mathsf{ct}), \sigma)$ as a forgery.

This completes the description of $\mathcal{B}$. In the following, we show that $\mathcal{B}$ breaks the unforgeability of Π_{CS}. Let $\mathsf{VK}_{[0,\ell]} := (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$. Since we assume $\mathcal{A}$ breaks the soundness of Π'_{NIZK},

$$\Pr[x^* \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, (\mathsf{VK}_{[0,\ell]}, Y^*), \pi) = \top]$$

is non-negligible where $Y^* = \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i^*}\}_{i\in[n+|\mathsf{ct}|]})$ and $y^* := (x^*, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$. On the other hand, by the extractability of Π_{NIZK},

$$\Pr[\mathsf{Verify}(\mathsf{crs}, (\mathsf{VK}_{[0,\ell]}, Y^*), \pi) = \top \wedge ((\mathsf{VK}_{[0,\ell]}, Y^*), (K, \sigma)) \notin \widetilde{\mathcal{R}}]$$

is negligible. Therefore

$$\Pr[x^* \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, (\mathsf{VK}_{[0,\ell]}, Y^*), \pi) = \top \wedge ((\mathsf{VK}_{[0,\ell]}, Y^*), (K, \sigma)) \in \widetilde{\mathcal{R}}]$$

is non-negligible. Suppose that this event happens. Since we have $x^* \notin \mathcal{L}$, we have $f_{\mathsf{pp}_{\mathsf{SKE}}}(K, x^*, \mathsf{ct}) = 0$. On the other hand, $((\mathsf{VK}_{[0,\ell]}, Y^*), (K, \sigma)) \in \widetilde{\mathcal{R}}$ implies that we have $K \in \{0,1\}^\ell \wedge \mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top$ where $Z = \mathsf{CS.Aggrgt}\big(\mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,K_i}\}_{i\in[\ell]}), Y^*\big)$, which implies $\mathsf{CS.Vrfy}(\mathsf{vk}, (K, x^*, \mathsf{ct}), \sigma) = \top$. This means that $\mathcal{B}$ succeeds in breaking the unforgeability of Π_{CS}. □

Theorem 5.2 (Zero-Knowledge). *The above NIZK scheme Π'_{NIZK} is computationally zero-knowledge if Π_{NIZK} is computationally zero-knowledge and Π_{SKE} is one-time secure.*

Proof. Let $(\mathcal{S}_1, \mathcal{S}_2)$ be the simulator for Π_{NIZK}. We describe the simulator $(\mathcal{S}'_1, \mathcal{S}'_2)$ for Π'_{NIZK} below.

$\mathcal{S}'_1(1^\kappa)$**:** It generates $(\mathsf{crs}, \tau_{\mathsf{V}}) \xleftarrow{\$} \mathcal{S}_1(1^\kappa)$, $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$, $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell+n+|\mathsf{ct}|],b\in\{0,1\}}), \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^{\ell+n+|\mathsf{ct}|})$, and $\mathsf{sk}_f \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, f_{\mathsf{pp}_{\mathsf{SKE}}})$, and outputs $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk}, \mathsf{sk}_f)$ and $\tau'_{\mathsf{V}} := \tau_{\mathsf{V}}$.

$\mathcal{S}'_2(\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk}, \mathsf{sk}_f), \tau'_{\mathsf{V}} = \tau_{\mathsf{V}}, x)$**:** It generates $K \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(K, 0^m)$, $Y := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i}\}_{i\in[n+|\mathsf{ct}|]})$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$, and $\pi \xleftarrow{\$} \mathcal{S}_2(\mathsf{crs}, \tau_{\mathsf{V}}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y))$, and outputs $\pi' := (\mathsf{ct}, \pi)$.

This completes the description of the simulator. We prove that proofs simulated by the above simulator are computationally indistinguishable from the honestly generated proofs. To prove this, we consider the following sequence of games between a PPT adversary $\mathcal{A}$ and a challenger.

G_0**:** In this game, proofs are generated honestly. Namely,

1. The challenger generates $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$, $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$, $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell+n+|\mathsf{ct}|],b\in\{0,1\}})$, $\mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^{\ell+n+|\mathsf{ct}|})$, and $\mathsf{sk}_f \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, f_{\mathsf{pp}_{\mathsf{SKE}}})$, and gives $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp}_{\mathsf{SKE}}, \mathsf{vk}, \mathsf{sk}_f)$ to $\mathcal{A}$.
2. $\mathcal{A}$ is given $(1^\kappa, \mathsf{crs}')$ and is allowed to query $\mathcal{O}(\mathsf{crs}', \cdot, \cdot)$, which works as follows. When $\mathcal{A}$ queries (x, w), if $(x, w) \notin \mathcal{R}$, then the oracle returns $\bot$. Otherwise, it picks $K \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, K, w)$, $\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_f, (K, x, \mathsf{ct}))$, $Y := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{\ell+i,y_i}\}_{i\in[n+|\mathsf{ct}|]})$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$, and $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y), (K, \sigma))$, and returns a proof $\pi' := (\mathsf{ct}, \pi)$.
3. Finally, $\mathcal{A}$ returns a bit β.

G_1**:** This game is identical to the previous game except that crs and π are generated differently. Namely, the challenger generates $(\mathsf{crs}, \tau_{\mathsf{V}}) \xleftarrow{\$} \mathcal{S}_1(1^\kappa)$ at the beginning of the game, and π is generated as $\pi \xleftarrow{\$} \mathcal{S}_2(\mathsf{crs}, \tau_{\mathsf{V}}, (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Y))$ for each oracle query.

G_2**:** This game is identical to the previous game except that ct is generated as $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, K, 0^m)$ for each oracle query.

Let T_i be the event that $\mathcal{A}$ returns 1 in G_i for $i = 0, 1, 2$. It is easy to see that proofs are generated by $\mathcal{S}' = (\mathcal{S}'_1, \mathcal{S}'_2)$ in G_2. Thus we have to prove that $|\Pr[\mathsf{T}_0] - \Pr[\mathsf{T}_2]|$ is negligible. The following lemmas are straightforward to prove.

Lemma 5.3. *If* Π_{NIZK} *satisfies computational zero-knowledge w.r.t. the simulator* $\mathcal{S}$*, then* $|\Pr[\mathsf{T}_0] - \Pr[\mathsf{T}_1]| = \mathsf{negl}(\kappa)$.

Proof. We observe that every proof π given to $\mathcal{A}$ is created for a correct statement in both games. Therefore, the indistinguishability of the games can be reduced to the zero-knowledge property of Π_{NIZK}. □

Lemma 5.4. *If* Π_{SKE} *is one-time secure, then* $|\Pr[\mathsf{T}_1] - \Pr[\mathsf{T}_2]| = \mathsf{negl}(\kappa)$.

Proof. Due to the change we introduced in G_1, the secret key K of SKE that is used to generate ct is not used anywhere else in both games. Moreover, each K is used only once. Therefore, the indistinguishability of these games can be reduced to the one-time security of Π_{SKE}. □

This completes the proof of Theorem 5.2. □

Remark 5.5. For instantiating this construction, we need to assume the MDDH assumption in a pairing group and the CDH assumption in a subgroup of $\mathbb{Z}_p^*$. The latter assumption is needed for constructing an SKE scheme with the required properties, which we do not know if we can construct in a pairing group. In more detail, for constructing such an SKE scheme, we need to assume the CDH assumption in a group in which a multiple product can be computed in $\mathbf{NC}^1$. (See [KNYY19b] for details.) While that is known to be possible in a subgroup of $\mathbb{Z}_p^*$ [BCH86], we are unaware of a similar result for a pairing group. Thus, we assume the CDH assumption in a subgroup of $\mathbb{Z}_p^*$ as an additional assumption.

5.2 Perfect Zero-Knowledge Variant

Here, we give a variant of our NIZK given in Section 5.1 that satisfies perfect zero-knowledge. The construction supports relations that are verified in $\mathbf{NC}^1$ and the proof size is $|w| \cdot \mathsf{poly}(\kappa)$. This is the first pairing-based construction of a NIZK with perfect zero-knowledge for a non-trivial calss of languages whose proof size is independent of the size of the circuit that verifies the corresponding relation.

Dual-Mode NIZK. Before stating our construction, we define the notion of dual-mode NIZK, which is used as a building block of our construction.

Definition 5.6 (Dual-Mode NIZK). *A dual-mode NIZK for a language* $\mathcal{L}$ *corresponding to a relation* $\mathcal{R}$ *consists of* 6 *PPT algorithms* $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}} = (\mathsf{SimSetup}, \mathsf{ExtSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}, \mathsf{Extract})$.

$\mathsf{SimSetup}(1^\kappa) \to (\mathsf{crs}, \tau_{\mathsf{Sim}})$: *The setup algorithm in the hiding mode takes as input the security parameter* 1^κ *and outputs a common reference string* crs *and a simulation trapdoor* τ_{Sim}.

$\mathsf{ExtSetup}(1^\kappa) \to (\mathsf{crs}, \tau_{\mathsf{Extract}})$: *The setup algorithm in the binding mode takes as input the security parameter* 1^κ *and outputs a common reference string* crs *and a extraction trapdoor* τ_{Extract}.

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

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

$\mathsf{Sim} : (\tau_{\mathsf{Sim}}, x) \to \pi$: *The simulation algorithm takes as input a simulation trapdoor* τ_{Sim} *and a statement* x *and outputs a simulated proof* π.

$\mathsf{Extract}(\tau_{\mathsf{Extract}}, \pi) \to w$: *The extraction algorithm takes as input an extraction trapdoor* τ_{Extract} *and a proof* π *and outputs an extracted witness* w.

A NIZK proof Π_{NIZK} *must satisfy the following requirements for all* $\kappa \in \mathbb{N}$*, where the probabilities are taken over the random choice of the algorithms.*

Completeness. *For all pairs* $(x, w) \in \mathcal{R}$*, if we run* $(\mathsf{crs}, \tau_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\kappa)$ *(or* $(\mathsf{crs}, \tau_{\mathsf{Extract}}) \xleftarrow{\$} \mathsf{ExtSetup}(1^\kappa)$*), then we have*

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

Perfect Simulation in Hiding Mode. *For all unbounded-time adversaries* $\mathcal{A}$*, if we run* $(\mathsf{crs}, \tau_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\kappa)$*, then we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs},\cdot,\cdot)}(1^\kappa, \mathsf{crs}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\tau_{\mathsf{Sim}},\cdot,\cdot)}(1^\kappa, \mathsf{crs}) = 1]\right| = 0,$$

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

Perfect Extraction in Binding Mode. *If we run* $(\mathsf{crs}, \tau_{\mathsf{Extract}}) \xleftarrow{\$} \mathsf{ExtSetup}(1^\kappa)$*, then for all* x *and* π *such that* $\mathsf{Verify}(\mathsf{crs}, x, \pi) = \top$*, we have*

$$\Pr[w \xleftarrow{\$} \mathsf{Extract}(\tau_{\mathsf{Extract}}, \pi) : (x, w) \in \mathcal{R}] = 1.$$

(Non-Uniform) Mode Indistinguishability. *For all non-uniform polynomial-time adversaries* $\mathcal{A}$*,* [10] *we have*

$$\left|\Pr[(\mathsf{crs}, \tau_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1] - \Pr[(\mathsf{crs}, \tau_{\mathsf{Extract}}) \xleftarrow{\$} \mathsf{Extract}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1]\right| = \mathsf{negl}(\kappa),$$

Lemma 5.7 ([GOS12, GS12]). *There exists a dual-mode NIZK under the DLIN or SXDH assumption.*

Construction. Now, we describe our construction of a NIZK with perfect zero-knowledge. 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 $(x, w) \in \mathcal{R}$.
The construction will be given by combining following ingredients.

- A constrained signature scheme $(\mathsf{CS.Setup}, \mathsf{CS.KeyGen}, \mathsf{CS.Sign}, \mathsf{CS.Vrfy}, \mathsf{CS.Aggrgt}, \mathsf{CS.VrfyOnL})$ we constructed in Section 4. The scheme should support the circuit C.

[10] We need security against non-uniform adversaries in the proof of soundness in our NIZK with perfect zero-knowledge.

- (Not necessarily compact) dual-mode NIZK scheme $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}} = (\mathsf{SimSetup}, \mathsf{ExtSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}, \mathsf{Extract})$ for the language $\widetilde{\mathcal{L}}$ corresponding to the relation $\widetilde{\mathcal{R}}$ defined below:
$((\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[m],b\in\{0,1\}}, X), (w, \sigma)) \in \widetilde{\mathcal{R}}$ if and only if the followings are satisfied:
  1. $w \in \{0,1\}^m$,
  2. $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top$ where $Z = \mathsf{CS.Aggrgt}\big(X, \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,w_i}\}_{i\in[m]})\big)$

Our NIZK with perfect zero-knowledge $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}} = (\mathsf{Setup}^{\mathsf{pzk}}, \mathsf{Prove}^{\mathsf{pzk}}, \mathsf{Verify}^{\mathsf{pzk}})$ is described as follows.

$\mathsf{Setup}^{\mathsf{pzk}}(1^\kappa)$**:**

1. Generate $(\mathsf{crs}, \tau_{\mathcal{S}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\kappa)$.
2. Generate $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}}), \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^{n+m})$.
3. Generate $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$.
4. Output $\mathsf{crs}' = (\mathsf{crs}, \mathsf{vk}, \mathsf{sk}_C)$.

$\mathsf{Prove}^{\mathsf{pzk}}(\mathsf{crs}', x, w)$**:**

1. Abort if $\mathcal{R}(x, w) = 0$. Otherwise, do the following.
2. Parse $\mathsf{crs}' \to (\mathsf{crs}, \mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}}), \mathsf{sk}_C)$.
3. Compute $\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_C, (x, w))$.
4. Compute $X := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$.
5. Compute $\pi \xleftarrow{\$} \mathsf{Prove}((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X), (w, \sigma))$.
6. Output π.

$\mathsf{Verify}^{\mathsf{pzk}}(\mathsf{crs}', x, \pi)$**:**

1. Parse $\mathsf{crs}' \to (\mathsf{crs}, \mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}})$.
2. Compute $X := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$.
3. Output $\top$ if $\mathsf{Verify}((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X), \pi) = \top$ and otherwise $\bot$.

Correctness. Suppose that π is an honestly generated proof on $(x, w) \in \mathcal{R}$. Then we have $\pi \xleftarrow{\$} \mathsf{Prove}((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X), (w, \sigma))$ where $\sigma \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_C, (x, w))$ and $X = \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$. By the correctness of Π_{CS}, we have $\mathsf{CS.Vrfy}(\mathsf{vk}, (x, w), \sigma) = \top$, which is equivalent to

$$\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top \quad \text{where} \quad Z = \mathsf{CS.Aggrgt}\big(X, \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{n+i,w_i}\}_{i\in[m]})\big).$$

Therefore we have $((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X), (w, \sigma)) \in \widetilde{\mathcal{R}}$ and thus $\mathsf{Verify}((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, Z), \pi) = \top$ by the correctness of $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$.

Efficiency. Since the relation $\widetilde{\mathcal{R}}$ which is proven by using Π_{NIZK} can be verified by a $\mathsf{poly}(\kappa, |w|)$-sized circuit independently of x or $|C|$, the size of π is $\mathsf{poly}(\kappa, |w|)$ and independent of x or $|C|$. Similarly to the scheme in Section 5.1, since the relation $\widetilde{\mathcal{R}}$ is well-suited to be proven by the Groth-Sahai proof, if we instantiate the scheme based on the Groth-Sahai proof, then the protocol is quite efficient, and the proof size can be made linear in the witness size, i.e., $|\pi| = |w| \cdot \mathsf{poly}(\kappa)$. More precisely, by a similar calculation to the one done in Appendix B, a proof consists of $6|w| + 14$ elements of $\mathbb{G}_1$ and $6|w| + 24$ elements of $\mathbb{G}_2$ when instantiated under the SXDH assumption. We note that it is not useful to extend the scheme to support all NP languages by expanding the witness since this would result in a scheme with proof size $|C| \cdot \mathsf{poly}(\kappa)$ which is no better than the Groth-Ostrovsky-Sahai NIZK.

Security. In the following, we prove the soundness and the perfect zero-knowledge property of $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}}$.

Theorem 5.8 (Soundness). *The above NIZK scheme* $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}}$ *satisfies computational (non-adaptive) soundness if* $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$ *satisfies perfect extractability in the binding mode, non-uniform mode-indistinguishability, and* Π_{CS} *is non-uniformly unforgeable.*

Proof. Suppose that $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}}$ does not satisfy computational non-adaptive soundness. Then there exists $x^* \notin \mathcal{L}$ and a non-uniform polynomial-time adversary $\mathcal{A}$ such that if we generate $\mathsf{crs}' = (\mathsf{crs}, \mathsf{vk}, \mathsf{sk}_C) \xleftarrow{\$} \mathsf{Setup}^{\mathsf{pzk}}(1^\kappa)$ and $\pi^* \xleftarrow{\$} \mathcal{A}(\mathsf{crs}')$, then $\Pr[\mathsf{Verify}^{\mathsf{pzk}}(\mathsf{crs}', x^*, \pi^*) = \top]$ is non-negligible. By the non-uniform mode-indistinguishability of $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$, if we generate crs by $\mathsf{ExtSetup}(1^\kappa)$ instead of $\mathsf{SimSetup}(1^\kappa)$, $\Pr[\mathsf{Verify}^{\mathsf{pzk}}(\mathsf{crs}', x^*, \pi^*) = \top]$ is still non-negligible. Then we construct a PPT adversary $\mathcal{B}$ that breaks the unforgeability of Π_{CS} as follows.

$\mathcal{B}(\mathsf{vk})$**:** It queries C to the key generation oracle to obtain sk_C. Then it generates $(\mathsf{crs}, \tau_{\mathsf{Ext}}) \xleftarrow{\$} \mathsf{ExtSetup}(1^\kappa)$, runs $\mathcal{A}(\mathsf{crs}')$ to obtain π^* where $\mathsf{crs}' := (\mathsf{crs}, \mathsf{vk}, \mathsf{sk}_C)$. Then it computes $(w, \sigma) \xleftarrow{\$} \mathsf{Extract}(\tau_{\mathsf{Ext}}, \pi^*)$ and outputs $((x^*, w), \sigma)$ as a forgery.

This completes the description of $\mathcal{B}$. In the following, we show that $\mathcal{B}$ breaks the unforgeability of Π_{CS}. By the perfect extractability of $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$ in the binding mode, we have $((\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[m],b\in\{0,1\}}, X), (w, \sigma)) \in \widetilde{\mathcal{R}}$ whenever $\mathsf{Verify}^{\mathsf{pzk}}(\mathsf{crs}', x^*, \pi^*) = \top$, which happens with non-negligible probability. In this case, we have $w \in \{0,1\}^m \wedge \mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma) = \top$ where $Z = \mathsf{CS.Aggrgt}\big(X, \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{n+i,w_i}\}_{i\in[m]})\big)$, which implies $\mathsf{CS.Vrfy}(\mathsf{vk}, (x^*, w), \sigma) = \top$. On the other hand, we never have $(x^*, w) \in \mathcal{R}$ for any $w \in \{0,1\}^m$ since $x^* \notin \mathcal{L}$ and thus we have $C(x^*, w) = 0$. This means that $\mathcal{B}$ succeeds in breaking the unforgeability of Π_{CS}. This completes the proof of Theorem 5.8. □

Remark 5.9. In the above proof, we assumed non-uniform security for underlying primitives since we rely on non-uniform reduction algorithms into which x^* is hardwired.

Remark 5.10. It is not clear how to extend the above proof to the case of the adaptive soundness. The problem is that the adversary's advantage may non-negligibly differs when changing crs from hiding mode to binding mode even under the mode-indistinguishability since the winning condition of the adaptive soundness is not efficiently verifiable. Indeed, Pass [Pas13] ruled out a black-box reduction from the adaptive soundness of a NIZK with statistical (and thus also perfect) zero-knowledge to any falsifiable assumption. On the other hand, we can prove a relaxed variant of the adaptive soundness called the *adpative culpable soundness* similarly to [GOS12]. Intuitively, the adaptive culpable soundness ensures the target statement x^* as long as it provides a witness for $x^* \notin \mathcal{L}$. With this relaxed definition, the winning condition is efficiently verifiable, and the above problem does not occur. In this sense, our scheme satisfies essentially the same level of soundness under the same assumption as that of Groth-Ostrovsky-Sahai proof [GOS12].

Theorem 5.11 (Perfect Zero-Knowledge). *The above NIZK scheme* $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}}$ *is perfectly zero-knowledge if* $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$ *is perfectly zero-knowledge in the hiding mode.*

Proof. We describe the simulator $(\mathcal{S}'_1, \mathcal{S}'_2)$ for $\Pi^{\mathsf{pzk}}_{\mathsf{NIZK}}$ below.

$\mathcal{S}'_1(1^\kappa)$**:** It generates $(\mathsf{crs}, \tau_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\kappa)$, $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}}), \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa, 1^{n+m})$, and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$, and outputs $\mathsf{crs}' := (\mathsf{crs}, \mathsf{vk}, \mathsf{sk}_C)$ and $\tau'_{\mathsf{V}} := \tau_{\mathsf{Sim}}$.

$\mathcal{S}'_2(\mathsf{crs}' := (\mathsf{crs}, \mathsf{vk}, \mathsf{sk}_C), \tau'_{\mathsf{V}} = \tau_{\mathsf{Sim}}, x)$**:** It first computes $X := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$. It then computes $\pi \xleftarrow{\$} \mathsf{Sim}(\tau_{\mathsf{Sim}}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X))$ and outputs π.

This completes the description of the simulator. Since we have $(\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X) \in \widetilde{\mathcal{L}}$ whenever $x \in \mathcal{L}$ by the correctness of Π_{CS}, the proof generated by the above simulator is perfectly indistinguishable from the real one by the perfect zero-knowledgeness of $\Pi^{\mathsf{dm}}_{\mathsf{NIZK}}$ in the hiding mode. This completes the proof of Theorem 5.11. □

6 UC-NIZK

6.1 UC Framework

Here, we briefly recall the UC framework. We refer to [Can01] for the full descriptions. The following description is taken verbatim from [KNYY19b]. Readers familiar with the UC framework can safely skip this section.

The UC framework. The UC-security is formalized by indistinguishability of real and ideal worlds. In the real world, parties $P_1, ..., P_N$ execute a protocol Π, and an adversary $\mathcal{A}$ may corrupt some of them. We say that $\mathcal{A}$ is adaptive if it adaptively decides which party to corrupt. In the ideal world, dummy parties $\tilde{P}_1, ..., \tilde{P}_N$ execute an ideal functionality $\mathcal{F}$, and a simulator $\mathcal{S}$ may corrupt some of them. In addition to (dummy) parties and an adversary/simulator, we consider another entity $\mathcal{Z}$ which tries to distinguish these two worlds. In the real (resp. ideal) world, an environment $\mathcal{Z}$, which takes the security parameter 1^κ and an auxiliary input z as input, can arbitrarily interact with $\mathcal{A}$ (resp. $\mathcal{S}$), and it can also feed any input to any uncorrupted party (resp. dummy party) to let it honestly run the protocol Π (resp. the ideal functionality) on the input and report the output to $\mathcal{Z}$. Finally, $\mathcal{Z}$ outputs a bit as its guess of in which world it is. We denote the output distribution of $\mathcal{Z}$ with auxiliary input z in the real world by $\mathtt{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\kappa, z)$, and denote the ensemble $\{\mathtt{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ by $\mathtt{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ Similarly, we denote the output distribution of $\mathcal{Z}$ with auxiliary input z in the ideal world by $\mathtt{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\kappa, z)$, and denote the ensemble $\{\mathtt{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ by $\mathtt{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

Definition 6.1. *Let $\mathcal{X} = \{X(1^\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ and $\mathcal{Y} = \{Y(1^\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$ be two distribution ensembles over $\{0,1\}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are indistinguishable (denoted by $\mathcal{X} \approx \mathcal{Y}$) if for any $c, d \in \mathbb{N}$, there exists $\kappa_0 \in \mathbb{N}$ such that we have $|\Pr[X(1^\kappa, z) = 1] - \Pr[Y(1^\kappa, z) = 1]| < \kappa^{-c}$ for all $\kappa > \kappa_0$ and all $z \in \{0,1\}^{\leq\kappa^d}$.*

Definition 6.2. *We say that Π UC-realizes $\mathcal{F}$ if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for all PPT environment $\mathcal{Z}$ [11], we have $\mathtt{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathtt{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.*

Hybrid models. We often construct a protocol in a setting where (multiple copies of) an ideal functionality $\mathcal{F}$ is available for each party. We call such a model $\mathcal{F}$-hybrid model. Definition 6.2 can be extended to define the notion of a protocol Π securely realizing a functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model.

Compositions and universal composition theorem. For a protocol ρ in the $\mathcal{F}$-hybrid model, and a protocol Π that realizes $\mathcal{F}$ (in the standard model), we can naturally define a composed protocol ρ^Π in the standard model, in which calls for $\mathcal{F}$ by ρ are responded by Π instead of $\mathcal{F}$. Canetti [Can01] proved the following *universal composition theorem*.

Theorem 6.3 ([Can01]). *If ρ UC-realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model and Π UC-realizes $\mathcal{F}$, then ρ^Π UC-realizes $\mathcal{G}$.*

Remark 6.4 (Static/Adaptive Security and Erasure). In some works of the UC-security, one considers a weaker security called the *static* security, which only considers adversaries that declare which party to corrupt at the beginning of the experiment. Also, one often considers the adaptive UC-security in the setting where parties can securely erase their internal state information so that an adversary that later corrupts the party cannot see the erased information. In this paper, we consider the strongest UC-security against *adaptive* adversaries *without assuming secure erasures*.

6.2 Ideal Functionalities.

Here, we recall ideal functionalities of CRS and NIZK. The ideal functionality of CRS denoted by $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ is given in Figure 3, and that of NIZK denoted by $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ is given in Figure 4. The descriptions here are taken verbatim from [CsW19]. The ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ captures correctness, soundness, and zero-knowledge as roughly explained below:

- Correctness is captured since all proofs generated by $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ are stored and thus accepted.

[11] Strictly speaking, $\mathcal{Z}$ is limited to be a *balanced* one, which roughly means that $\mathcal{Z}$ should not give too much inputs to the adversary compared to inputs given to the other parties. See [Can00] for more details.

Ideal Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ proceeds as follows, running with parties $P_1, ..., P_N$ and an adversary $\mathcal{S}$, and parametrized by a distribution $\mathcal{D}$.

- Upon receiving a message $(\mathsf{init}, \mathsf{sid})$ from party P_i, do:
  1. If there is no value $(\mathsf{sid}, \mathsf{crs})$ recorded, then sample $\mathsf{crs} \xleftarrow{\$} \mathcal{D}$ and record it.
  2. Send $(\mathsf{sid}, \mathsf{crs})$ as a public delayed output to P_i.[a]

[a] "Public delayed output" means that the value is first sent to $\mathcal{S}$, and then sent to P_i after the permission by $\mathcal{S}$. See [Can01] for details.

Figure 3: The common reference string functionality

Ideal Functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$

$\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ proceeds as follows, running with parties $P_1, ..., P_N$ and an adversary $\mathcal{S}$, and parametrized by a relation $\mathcal{R}$.

Proof: On input $(\mathsf{prove}, \mathsf{sid}, x, w)$ from a party P_i, ignore if $(x, w) \notin \mathcal{R}$. Send $(\mathsf{prove}, P_i, \mathsf{sid}, x)$ to $\mathcal{S}$ and wait for $(\mathsf{proof}, \mathsf{sid}, \pi)$. Upon receiving the answer store (sid, x, π) and send $(\mathsf{proof}, \mathsf{sid}, \pi)$ to P_i.

Verification: On input $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ from a party P_j check whether (sid, x, π) is stored. If not send $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$ to $\mathcal{S}$ and wait for an answer $(\mathsf{witness}, w)$. Upon receiving the answer, check whether $(x, w) \in \mathcal{R}$ and in that case, store (sid, x, π). If (sid, x, π) has been stored return $(\mathsf{verification}, \mathsf{sid}, x, \pi, \top)$ to P_j, else return $(\mathsf{verification}, \mathsf{sid}, x, \pi, \bot)$ to P_j.

Figure 4: The NIZK functionality

- Soundness is captured since if a proof π on a statement x that has not been generated by $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ passes the verification, then $\mathcal{S}$ should succeed in extracting a valid witness w for the statement x, which is possible only when $x \in \mathcal{L}$ where $\mathcal{L} = \{x | \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$.
- Zero-knowledge is captured since $\mathcal{S}$ generates a proof π without knowing a witness w.

6.3 Construction

Here, we give our construction of an adaptive UC-NIZK scheme. Let $\mathcal{R}$ be any relation over $\{0,1\}^n \times \{0,1\}^m$ verified by a circuit C (i.e., we have $C(x, w) = 1$ if and only if $(x, w) \in \mathcal{R}$ and 0 otherwise). We construct a UC-NIZK protocol in the $(\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}, \mathcal{F}_{\mathsf{crs}}^{\mathcal{D}_{\mathsf{CS}}})$-hybrid model based on the following building blocks where $\widetilde{\mathcal{R}}$, $\mathcal{D}_{\mathsf{CS}}$ and $\mathcal{D}_{\mathsf{EQCom}}$ are defined below:

- A constrained signature scheme $\mathsf{CS} = (\mathsf{CS.Setup}, \mathsf{CS.KeyGen}, \mathsf{CS.Sign}, \mathsf{CS.Vrfy}, \mathsf{CS.Aggrgt}, \mathsf{CS.VrfyOnL})$ with decomposable online-offline efficiency as defined in Definition 4.2. The distribution $\mathcal{D}_{\mathsf{CS}}$ is the distribution of $(\mathsf{vk}, \mathsf{sk}_C)$ where $(\mathsf{vk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa)$, $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$.
- A strongly unforgeable OTS scheme $\mathsf{OTS} = (\mathsf{OTS.KeyGen}, \mathsf{OTS.Sign}, \mathsf{OTS.Verify})$.
- The relation $\widetilde{R}$ is defined as follows:
  $((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), (w, \sigma_{\mathsf{CS}})) \in \widetilde{\mathcal{R}}$ if and only if the followings are satisfied:
  1. $w \in \{0,1\}^m$
  2. $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma_{\mathsf{CS}}) = \top$ where $Z = \mathsf{CS.Aggrgt}(X, \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{n+i,w_i}\}_{i\in[m]}))$

Our UC-NIZK scheme Π_{UCNIZK} in the $(\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}, \mathcal{F}_{\mathsf{crs}}^{\mathcal{D}_{\mathsf{CS}}})$-hybrid model is described in Figure 5. Here, we remark that we describe the scheme as if there is a common reference string available for every party. Strictly speaking, we should

Protocol Π_{UCNIZK}

- Common reference string: $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}}), \mathsf{sk}_C)$ where $(\mathsf{vk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa)$ and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$.
- Upon receiving $(\mathsf{prove}, \mathsf{sid}, x, w)$, a party proceeds as follows:
  1. Compute $\sigma_{\mathsf{CS}} \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_C, (x, w))$
  2. Compute $X \coloneqq \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$.
  3. Generate $(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{sigk}_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^\kappa)$.
  4. Send $(\mathsf{prove}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), (w, \sigma_{\mathsf{CS}}))$ to $\mathcal{F}^{\widetilde{\mathcal{R}}}_{\mathsf{NIZK}}$.
  5. Wait for the answer $(\mathsf{proof}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}})$.
  6. Compute $\sigma_{\mathsf{OTS}} \xleftarrow{\$} \mathsf{OTS.Sign}(\mathsf{sigk}_{\mathsf{OTS}}, (x, \pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}))$.
  7. Return $(\mathsf{proof}, \mathsf{sid}, x, (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}}))$.
- Upon receiving $(\mathsf{verify}, \mathsf{sid}, x, \pi)$, a party proceeds as follows:
  1. Parse π as $(\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}})$.
  2. Verify that $\mathsf{OTS.Verify}(\mathsf{vk}_{\mathsf{OTS}}, (x, \pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}), \sigma_{\mathsf{OTS}}) = \top$. If not return $(\mathsf{verification}, \mathsf{sid}, x, \pi_{\mathsf{NIZK}}, \bot)$.
  3. Compute $X \coloneqq \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$.
  4. Send $(\mathsf{verify}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}})$ to $\mathcal{F}^{\widetilde{\mathcal{R}}}_{\mathsf{NIZK}}$.
  5. Wait for the answer $(\mathsf{verification}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}}, b)$.
  6. Return $(\mathsf{verification}, \mathsf{sid}, x, \pi, b)$.

Figure 5: Adaptively secure UC-NIZK protocol

implement this by using the ideal functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}_{\mathsf{CS}}}$. For notational simplicity, we omit this, and just think that a common reference string is chosen at the beginning of the protocol execution and published for every party.

Security. The following theorem asserts the security of Π_{UCNIZK}.

Theorem 6.5. Π_{UCNIZK} *UC-realizes* $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ *in the* $(\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}, \mathcal{F}_{\mathsf{crs}}^{\mathcal{D}_{\mathsf{CS}}})$*-hybrid model tolerating adaptive, malicious adversaries.*

Proof. The proof follows a similar line to existing works that constructed UC-NIZKs [GGI+15, CsW19, KNYY19b]. We note that many parts of the proof are taken verbatim from [KNYY19b] though there are some technical differences since we rely on different primitives. Let $\mathcal{A}$ be any PPT adaptive adversary. What we have to do is to construct a simulator that interacts with dummy parties $\tilde{P}_1, ..., \tilde{P}_N$ and the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ such that no environment $\mathcal{Z}$ can distinguish the simulated execution from the real execution of $\mathcal{A}$ that interacts with real parties $P_1, ..., P_N$ who run the real protocol Π_{UCNIZK}. We first consider a simulator $\mathcal{S}_{\mathsf{Real}}$ that perfectly simulates the real execution by using an extended capability that it can know inputs to the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ and control its output. (Note that this is not allowed in the ideal world. We consider the execution of $\mathcal{S}_{\mathsf{Real}}$ just as a mental experiment.) Then we gradually modify the simulator without letting the environment notice it with a non-negligible advantage, and finally present a legitimate simulator $\mathcal{S}_{\mathsf{Sim}}$ in the ideal world. We consider the following sequence of simulators.

$\mathcal{S}_{\mathsf{Real}}$**:** As noted above, $\mathcal{S}_{\mathsf{Real}}$ can know inputs to the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ and control its output. The simulator $\mathcal{S}_{\mathsf{Real}}$ along with dummy parties $\tilde{P}_1, ..., \tilde{P}_N$ perfectly simulates the execution between $\mathcal{A}$ and $P_1, ...P_N$ who run Π_{UCNIZK} by using these capabilities. Specifically, it works as follows:

- To simulate the common reference string, $\mathcal{S}_{\mathsf{Real}}$ first generates $(\mathsf{vk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{CS.Setup}(1^\kappa)$ and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{CS.KeyGen}(\mathsf{msk}, C)$, and sets $(\mathsf{vk} = (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+m],b\in\{0,1\}}), \mathsf{sk}_C)$ as a common reference string used throughout the execution.
- When $\mathcal{S}_{\mathsf{Real}}$ receives $(\mathsf{prove}, P_i, \mathsf{sid}, x)$ from the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$, it must be the case that honest $\tilde{P}_i$ has sent $(\mathsf{prove}, \mathsf{sid}, x, w)$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ such that $(x, w) \in \mathcal{R}$. By using the capability to see the input $(\mathsf{prove}, \mathsf{sid}, x, w)$, $\mathcal{S}_{\mathsf{Real}}$ honestly runs the proving algorithm of Π_{UCNIZK} on input $(\mathsf{prove}, \mathsf{sid}, x, w)$ to generate a proof π, and returns $(\mathsf{prove}, \mathsf{sid}, \pi)$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$.
- When $\mathcal{S}_{\mathsf{Real}}$ receives $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$ from the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$, it must be the case that $\tilde{P}_j$ has sent $(\mathsf{prove}, \mathsf{sid}, x, \pi)$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ and π is not a proof that has been generated by the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. $\mathcal{S}_{\mathsf{Real}}$ honestly runs the protocol Π_{UCNIZK} on input $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ to generate $(\mathsf{verification}, \mathsf{sid}, x, \pi, b)$, and instructs $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ to return $(\mathsf{verification}, \mathsf{sid}, x, \pi, b)$ to $\tilde{P}_j$ by using the capability to control the output of $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. Note that $\mathcal{S}_{\mathsf{Real}}$ does not give a witness w to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. This is not needed since $\mathcal{S}_{\mathsf{Real}}$ has the capability to control the behavior of $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$.
- To simulate the interaction between $\mathcal{A}$ and $\mathcal{Z}$, $\mathcal{S}_{\mathsf{Real}}$ just internally simulates $\mathcal{A}$, and forwards all communications between $\mathcal{A}$ and $\mathcal{Z}$. Whenever $\mathcal{A}$ corrupts a party P_i, $\mathcal{S}_{\mathsf{Real}}$ corrupts the corresponding dummy party $\tilde{P}_i$, and simulate the communication between $\mathcal{A}$ and P_i. We note that since all proofs output by $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ are actually generated by $\mathcal{S}_{\mathsf{Real}}$, it knows all internal coins P_i is supposed to know. Therefore it can perfectly simulate the internal states of corrupted parties that are needed for simulating the communication between $\mathcal{A}$ and P_i.

Lemma 6.6. *If* Π_{CS} *and* Π_{OTS} *are correct, then we have* $\mathit{REAL}_{\Pi_{\mathsf{UCNIZK}},\mathcal{A},\mathcal{Z}} \approx \mathit{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Real}},\mathcal{Z}}$.

Proof. The only difference between the real execution of $\mathcal{A}$ interacting with $P_1, ..., P_N$ and the execution of $\mathcal{S}_{\mathsf{Real}}$ interacting with $\tilde{P}_1, ..., \tilde{P}_N$ from the view of $\mathcal{Z}$ is that in the latter, a proof generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ are always accepted by $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ (i.e., it returns $(\mathsf{verification}, \star, \star, \star, \top)$) without checking the validity of the proofs by using the actual verification protocol of Π_{UCNIZK}. On the other hand, it is easy to see that an honestly generated proof is accepted with probability 1 in Π_{UCNIZK} by the correctness of Π_{CS} and Π_{OTS} and the functionality of $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$. Therefore these two distributions are perfectly indistinguishable. □

$\mathcal{S}_{\mathsf{Ext}}$**:** This simulator works similarly to $\mathcal{S}_{\mathsf{Real}}$ except the way of simulating verification when it receives $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$ from the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. Unlike $\mathcal{S}_{\mathsf{Real}}$, $\mathcal{S}_{\mathsf{Ext}}$ does not use the power of controlling the behavior of $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. It then returns some $(\mathsf{witness}, w)$ whenever $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$ is sent from $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ as in the description of $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ as follows: Upon receiving $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$, $\mathcal{S}_{\mathsf{Ext}}$ honestly runs the verification algorithm of Π_{UCNIZK} on input $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ to obtain $(\mathsf{verification}, \mathsf{sid}, x, \pi, b)$. If $b = \bot$, then it returns $(\mathsf{witness}, \bot)$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. Otherwise, it parses $\pi = (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}})$, computes $X := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$, gives $(\mathsf{verification}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}})$ to $\mathcal{A}$, and waits for the response $(\mathsf{witness}, (w, \sigma_{\mathsf{CS}}))$ from $\mathcal{A}$. Then $\mathcal{S}_{\mathsf{Ext}}$ returns $(\mathsf{witness}, w)$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$.

Lemma 6.7. *If Π_{OTS} is strongly unforgeable and Π_{CS} is unforgeable, then we have $\mathit{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Real}},\mathcal{Z}} \approx \mathit{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Ext}},\mathcal{Z}}$.*

Proof. The differences between the executions of $\mathcal{S}_{\mathsf{Real}}$ and $\mathcal{S}_{\mathsf{Ext}}$ occurs only when an honest party sends $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ such that $(\mathsf{proof}, \mathsf{sid}, x, \pi)$ has not been generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$, π is a valid proof for the statement x, and $(x, w) \notin \mathcal{R}$ where w is the first component of the witness extracted from π_{NIZK} by $\mathcal{A}$. We prove that this happens with negligible probability. If $(\mathsf{proof}, \mathsf{sid}, x, \pi = (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}}))$ has not been generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$, there are the following two possible cases:

1. No proof of the form $(\mathsf{proof}, \mathsf{sid}, \star, (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \star))$ has been generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$.
2. A proof $(\mathsf{proof}, \mathsf{sid}, x', (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma'_{\mathsf{OTS}}))$ such that $(x', \sigma'_{\mathsf{OTS}}) \neq (x, \sigma_{\mathsf{OTS}})$ has been generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$.

In the second case, σ_{OTS} is a valid signature on the message $(x, \pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}})$ with negligible probability due to the strong one-time security of Π_{OTS}, and thus π is not a valid proof for the statement x. Since we are interested in the case that π is a valid proof for the statement x, we consider the first case in the following. If $\pi = (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}})$ is a valid proof for the statement x, then π_{NIZK} is a valid proof for the statement $(\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}})$ by the construction of Π_{UCNIZK} where $X = \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$. On the other hand, a proof $(\mathsf{proof}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}})$ has never been generated through $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$ since this happens only when a proof of the form $(\mathsf{proof}, \mathsf{sid}, \star, (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \star))$ is generated through $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. This means that $\mathcal{A}$ must succeed in extracting $(w, \sigma_{\mathsf{CS}})$ such that $((\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), (w, \sigma_{\mathsf{CS}})) \in \widetilde{\mathcal{R}}$ by the definition of ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$. Especially, we have $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, Z, \sigma_{\mathsf{CS}}) = \top$ where $Z = \mathsf{CS.Aggrgt}(X, \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{n+i,w_i}\}_{i\in[m]}))$, which implies $\mathsf{CS.Vrfy}(\mathsf{vk}, (x, w), \sigma_{\mathsf{CS}}) = \top$. Then we have $(x, w) \in \mathcal{R}$ except a negligible probability since otherwise we succeed in breaking the unforgeability of Π_{CS} by outputting $((x, w), \sigma_{\mathsf{CS}})$ as a forgery. (Remark that we have $C(x, w) = 0$ if $(x, w) \notin \mathcal{R}$.) In summary, a difference between executions of these two simulators occurs with negligible probability. □

$\mathcal{S}_{\mathsf{Sim}}$**:** This simulator works similarly to $\mathcal{S}_{\mathsf{Ext}}$ except the way of simulating a proof when it receives $(\mathsf{prove}, \mathsf{sid}, x)$ from the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$ and the way of simulating internal coins of corrupted parties.

- When it receives $(\mathsf{prove}, P_i, \mathsf{sid}, x)$, it computes $X := \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,x_i}\}_{i\in[n]})$, generates $(\mathsf{vk}_{\mathsf{OTS}}, \mathsf{sigk}_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^\kappa)$, and sends $(\mathsf{prove}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}))$ to $\mathcal{A}$, which returns $(\mathsf{proof}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}), \pi_{\mathsf{NIZK}})$. Then it computes $\sigma_{\mathsf{OTS}} \xleftarrow{\$} \mathsf{OTS.Sign}(\mathsf{sigk}_{\mathsf{OTS}}, (x, \pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}))$ and returns $(\mathsf{proof}, \mathsf{sid}, x, (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}}))$ to $\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}}$. $\mathcal{S}_{\mathsf{Sim}}$ stores all randomness used in the above simulation (i.e., randomness used in $\mathsf{OTS.KeyGen}$ and $\mathsf{OTS.Sign}$) along with the corresponding party P_i, session ID sid, the statement x, and the proof $\pi = (\pi_{\mathsf{NIZK}}, \mathsf{vk}_{\mathsf{OTS}}, \sigma_{\mathsf{OTS}})$.
- When the adversary $\mathcal{A}$ corrupts a party P_i, it corrupts $\tilde{P}_i$ who has generated k proofs $\pi_1, ..., \pi_k$. Then it knows from the internal state of $\tilde{P}_j$ that π_j was generated on an input (x_j, w_j) for each $j \in [k]$. For each $j \in [k]$, $\mathcal{S}_{\mathsf{Sim}}$ computes $\sigma_{\mathsf{CS},j} \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_C, (x_j, w_j))$. Now, $\mathcal{S}_{\mathsf{Sim}}$ knows the all internal information that is supposed to be held by P_i (i.e., w_j, $\sigma_{\mathsf{CS},j}$ and all randomness used in $\mathsf{OTS.KeyGen}$, $\mathsf{OTS.Sign}$, and $\mathsf{CS.Sign}$ for generating π_j and $\sigma_{\mathsf{CS},j}$). It uses this information to simulate the interaction between $\mathcal{A}$ and the corrupted party P_i.

Lemma 6.8. *We have $\mathit{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Ext}},\mathcal{Z}} \approx \mathit{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Sim}},\mathcal{Z}}$.*

Proof. The only difference between simulations by $\mathcal{S}_{\mathsf{Ext}}$ and $\mathcal{S}_{\mathsf{Sim}}$ is that $\mathcal{S}_{\mathsf{Ext}}$ generates π_{NIZK} through the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$ as specified in the proving algorithm of Π_{UCNIZK} whereas $\mathcal{S}_{\mathsf{Sim}}$ generates π_{NIZK} by sending $(\mathsf{prove}, \mathsf{sid}, (\mathsf{vk}_0, \{\mathsf{vk}_{n+i,b}\}_{i\in[m],b\in\{0,1\}}, X, \mathsf{vk}_{\mathsf{OTS}}))$ to the adversary $\mathcal{A}$. They are exactly identical from the view of the environment since the proving functionality of $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$ also generates π_{NIZK} in the latter way as is seen in the definition of $\mathcal{F}_{\mathsf{NIZK}}^{\widetilde{\mathcal{R}}}$. □

By combining the above lemmas, we have $\mathtt{REAL}_{\Pi_{\mathsf{UCNIZK}},\mathcal{A},\mathcal{Z}} \approx \mathtt{IDEAL}_{\mathcal{F}_{\mathsf{NIZK}}^{\mathcal{R}},\mathcal{S}_{\mathsf{Sim}},\mathcal{Z}}$. Here, we notice that $\mathcal{S}_{\mathsf{Sim}}$ no longer uses any extended capability, and it is a legitimate simulator in the ideal world. This concludes the proof of Theorem 6.5 □

Acknowledgement. We thank anonymous reviewers of Eurocrypt 2020 for their helpful comments. The first and the third authors were supported by JST CREST Grant Number JPMJCR19F6. The third author was supported by JSPS KAKENHI Grant Number 16K16068.

References

[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.)

[AHL+12] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theor. Comput. Sci.*, 422:15–38, 2012. (Cited on page 5.)

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Heidelberg, March 2011. (Cited on page 5.)

[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 29.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. (Cited on page 5.)

[BF11] 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 44.)

[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 1, 7.)

[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 28.)

[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 7.)

[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 7.)

[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 7.)

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067. (Cited on page 33.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Cited on page 33, 34.)

[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. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019. (Cited on page 7.)

[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 7.)

[CH85] Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14(4):833–839, 1985. (Cited on page 42, 44.)

[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.)

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 476–506. Springer, Heidelberg, November 2018. (Cited on page 7.)

[CsW19] Ran Cohen, abhi shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 30–60. Springer, Heidelberg, August 2019. (Cited on page 4, 33, 36.)

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014. (Cited on page 2.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. (Cited on page 8.)

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 601–620. Springer, Heidelberg, August 2015. (Cited on page 5, 12.)

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014. (Cited on page 5, 12.)

[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 3, 7.)

[Gen09] Craig Gentry. A fully homomorphic encryption scheme, 2009. Ph.D. thesis, Stanford University. (Cited on page 2.)

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, August 2013. (Cited on page 12.)

[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 27.)

[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, 3, 4, 36.)

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. (Cited on page 2, 7.)

[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 1.)

[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 3, 7.)

[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, 7, 30, 32.)

[Gro10a] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 341–358. Springer, Heidelberg, December 2010. (Cited on page 2, 3, 7.)

[Gro10b] 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, 7.)

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. (Cited on page 2.)

[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, 7, 30, 45.)

[GVW15] 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 42, 44, 45.)

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. (Cited on page 2.)

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, August 2016. (Cited on page 5, 12.)

[HW13] Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 162–179. Springer, Heidelberg, February / March 2013. (Cited on page 5.)

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002. (Cited on page 13.)

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017. (Cited on page 5, 12, 13.)

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 295–312. Springer, Heidelberg, August 2014. (Cited on page 2.)

[JR17] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. *Journal of Cryptology*, 30(4):1116–1156, October 2017. (Cited on page 2.)

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of Yao's garbled circuits. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 433–458. Springer, Heidelberg, October / November 2016. (Cited on page 5, 12.)

[KNYY19a] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 622–651. Springer, Heidelberg, May 2019. (Cited on page 2, 3, 4, 5, 7, 42.)

[KNYY19b] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Exploring constructions of compact NIZKs from various assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 639–669. Springer, Heidelberg, August 2019. (Cited on page 2, 3, 4, 7, 8, 9, 10, 27, 28, 29, 33, 36, 45.)

[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 7.)

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015. (Cited on page 2.)

[KW18] 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 3, 4.)

[KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019. (Cited on page 5, 11, 12, 13, 14, 15, 16, 19, 20.)

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. (Cited on page 2, 7.)

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer, Heidelberg, February 2011. (Cited on page 4.)

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. (Cited on page 2.)

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Li Gong and Michael K. Reiter, editors, *ACM CCS 98*, pages 59–66. ACM Press, November 1998. (Cited on page 2.)

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 35–52. Springer, Heidelberg, March 2011. (Cited on page 5.)

[Pas13] Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 334–354. Springer, Heidelberg, March 2013. (Cited on page 32.)

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019. (Cited on page 7.)

[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 7.)

[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 3, 42, 44.)

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). pages 196–203, 1976. (Cited on page 42.)

[VNS+03] V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 162–176. Springer, Heidelberg, December 2003. (Cited on page 13.)

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009. (Cited on page 7.)

A Homomorphic Signature from CS

Here, we give a construction of homomorphic signature (HomSig) scheme from CS. To do so, we apply the CS-to-HomSig conversion proposed by Tsabary [Tsa17]. Our version of the conversion here is slightly different from hers in that we sign on *circuits* rather than signing on *messages* (or data). We provide discussion on this difference in Remark A.4. We note that from a theoretical stand point, the two formalizations are equivalent (up to some small differences) since one can always view circuits and messages interchangeably using universal circuits [Val76, CH85].

A.1 Definition

In the following, we only define single-data HomSig for simplicity. The definition for the multi-data version can be found in [GVW15, KNYY19a].

Syntax. Let $\{\{0,1\}^{\ell(\kappa)}\}_{\kappa\in\mathbb{N}}$ 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 $\{0,1\}^{\ell(\kappa)}$ and range $\{0,1\}$.[12] Below, we omit the subscripts for simplicity.

Definition A.1 (Homomorphic Signatures). *A homomorphic signature (HomSig) scheme Π_{HS} with message space $\{0,1\}^\ell$ for the circuit class $\mathcal{C}$ is defined by the following five algorithms:*

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

[12] We can consider more general message and circuit classes, however, we consider this simplistic version to match the CS definition we provided.

$\mathsf{HS.Sign}(\mathsf{sk}, C) \to \boldsymbol{\sigma}$*: The signing algorithm takes as input a signing key* sk *and a circuit* $C \in \mathcal{C}$*, and outputs a signature* $\boldsymbol{\sigma}$*.*

$\mathsf{HS.Eval}(\mathsf{pk}, C, \mathbf{x}, \boldsymbol{\sigma}) \to \sigma$*: The signature-evaluation algorithm takes as input a verification key* pk*, a circuit* $C : \{0,1\}^\ell \to \{0,1\}$ *in* $\mathcal{C}$*, messages* $\mathbf{x} \in \{0,1\}^\ell$*, and a signature* $\boldsymbol{\sigma}$ *and outputs an evaluated signature* σ*.*

$\mathsf{HS.Verify}(\mathsf{pk}, \mathbf{x}, z, \sigma) \to \top$ *or* $\bot$*: The verification algorithm splits into a pair of algorithms* $(\mathsf{HS.VerifyOffL}, \mathsf{HS.VerifyOnL})$*:*

- $\mathsf{HS.VerifyOffL}(\mathsf{pk}, \mathbf{x}) \to \mathsf{pk}_{\mathbf{x}}$ *: The offline verification algorithm takes as input a verification key* pk*, messages* $\mathbf{x} \in \{0,1\}^\ell$*, and outputs an evaluated verification key* $\mathsf{pk}_{\mathbf{x}}$*.*
- $\mathsf{HS.VerifyOnL}(\mathsf{pk}', z, \sigma) \to \top$ *or* $\bot$ *: The online verification algorithm takes as input an (evaluated) verification key* pk'*, a message* $z \in \{0,1\}$*, and a evaluated signature* σ*, and outputs* $\top$ *if the signature is valid and outputs* $\bot$ *otherwise.*

Correctness. We define correctness for evaluated messages.

Definition A.2 (Correctness). *We say a homomorphic signature scheme* Π_{HS} *is correct, if for all* $\kappa \in \mathbb{N}$*,* $\ell \in \mathsf{poly}(\kappa)$*, messages* $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$*, circuits* $C \in \mathcal{C}$*, and* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{HS.KeyGen}(1^\kappa, 1^\ell)$ *we have*

$$\Pr[\mathsf{HS.Sign}(\mathsf{sk}, C) \to \boldsymbol{\sigma};\ \mathsf{HS.Eval}(\mathsf{pk}, C, \mathbf{x}, \boldsymbol{\sigma}) \to \sigma :\ \mathsf{HS.Verify}(\mathsf{pk}, \mathbf{x}, C(\mathbf{x}), \sigma) = \top] = 1,$$

where the probability is taken over the randomness used by all of the algorithms.

Unforgeability. We now define unforgeability for a HomSig scheme. 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 $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{HS.KeyGen}(1^\kappa, 1^\ell)$ and gives pk to $\mathcal{A}$.

Signing Query: The adversary $\mathcal{A}$ submits a circuit $C \in \mathcal{C}$ to be signed. The challenger responds by creating a signature $\boldsymbol{\sigma} \xleftarrow{\$} \mathsf{HS.Sign}(\mathsf{sk}, C)$ and sends $\boldsymbol{\sigma}$ to $\mathcal{A}$. Here, $\mathcal{A}$ can query a circuit only once.

Forgery: Then the adversary $\mathcal{A}$ outputs messages $\mathbf{x}^\star$, a message $z^\star \in \{0,1\}$, and a signature $\sigma^\star$ as the forgery. We say that $\mathcal{A}$ wins the game if:

1. $\mathbf{x}^\star \in \{0,1\}^\ell$; and
2. $C(\mathbf{x}^\star) \neq z^\star$; and
3. $\mathsf{HS.Verify}(\mathsf{pk}, \mathbf{x}^\star, z^\star, \sigma^\star) = \top$.

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.

Definition A.3 (Unforgeability). *A homomorphic signature scheme* Π_{HS} *is said to satisfy unforgeability if for any PPT adversary* $\mathcal{A}$ *the advantage* $\Pr[\mathcal{A} \text{ wins}]$ *of the above game is negligible.*

Compactness and Offline/Online Efficiency. We say a HomSig scheme Π_{HS} is *compact* if there exists a universal polynomial $\mathsf{poly}(\cdot)$ such that the evaluated signature σ (output by $\mathsf{HS.Eval}$) satisfies $|\sigma| \leq \mathsf{poly}(\kappa)$. Moreover, we say Π_{HS} is *online-offline efficient* if there exists a universal polynomial $\mathsf{poly}(\cdot)$ such that the running time of $\mathsf{HS.VerifyOnL}$ is $\mathsf{poly}(\kappa)$. Here, we allow the running time of the offline phase, i.e., running $\mathsf{HS.VerifyOffL}$, to depend on the size of the messages $|\mathbf{x}|$. In particular, once the messages $\mathbf{x}$ is fixed and we preprocess the evaluation key $\mathsf{pk}_{\mathbf{x}} \xleftarrow{\$} \mathsf{HS.VerifyOffL}(\mathsf{pk}, \mathbf{x})$, then we can verify an evaluated signature with respect to the messages $\mathbf{x}$ and any circuit C in time independent of $|C|$ or $|\mathbf{x}|$.

Remark A.4 (Comparison with HS signing on messages). It is common in the literatures to define the signing algorithm $\mathsf{HS.Sign}$ with respect to messages $\mathbf{x}$ rather than a circuit C, e.g., [BF11, GVW15]. In many natural applications, e.g., a client wants to outsource a computation of many algorithms on a fixed dataset, it may be more convenient to work with our formalization of HS. However, in either cases, we note that the two formalizations are equivalent from a theoretical point view in that both definitions are interchangeable using universal circuits. Namely, given messages $\mathbf{x}$, we can view it as a universal circuit $U(\cdot, \mathbf{x})$ which takes as input a circuit C (represented as say bit strings) output $C(\mathbf{x})$. Since Cook and Hoover [CH85] showed that $U(\cdot, \mathbf{x})$ can be only a constant times deeper than that of C, this conversion can be made without incurring any strong restrictions on the circuit class supported by the underlying HS. A shortcoming of this approach is that the signature size for the message $\mathbf{x}$ depends on the description size of C since the input-length of the universal circuit $U(\cdot, \mathbf{x})$ should be taken as large as the description size of C. On the other hand, compactness for an evaluated signature is preserved.

A.2 Construction of Homomorphic Signatures from CS

We construct a HomSig scheme with message space $\{0,1\}^\ell$ for circuit class $\mathcal{C} = \{C : \{0,1\}^\ell \to \{0,1\}\}$ from a CS scheme. Concretely, let $\Pi_{\mathsf{CS}} = (\mathsf{CS.Setup}, \mathsf{CS.KeyGen}, \mathsf{CS.Sign}, \mathsf{CS.Vrfy})$ be a CS scheme with message space $\{0,1\}^{\ell+1}$ for a circuit class $\tilde{\mathcal{C}}$ defined as

$$\tilde{\mathcal{C}} = \{\tilde{C} : \{0,1\}^{\ell+1} \to \{0,1\} \mid \text{for all } C \in \mathcal{C}, (\mathbf{x}, z) \in \{0,1\}^\ell \times \{0,1\}, \text{we have } \tilde{C}(\mathbf{x}, z) = \big(C(\mathbf{x}) \stackrel{?}{=} z\big)\}.$$

More specifically, $\tilde{\mathcal{C}}$ is a set of circuits with input length $\ell+1$ such that a circuit $\tilde{C} \in \tilde{\mathcal{C}}$ on input $(\mathbf{x}, z) \in \{0,1\}^\ell \times \{0,1\}$ outputs 1 if and only if the associated circuit $C \in \mathcal{C}$ outputs z on input $\mathbf{x}$. Since $(z' \stackrel{?}{=} z)$ can be expressed by a constant-sized circuit, every circuit in $\tilde{\mathcal{C}}$ has depth and size bounded by $d + O(1)$ and $s + O(1)$ respectively, where d and s denotes the maximal depth and size of the circuits in $\mathcal{C}$ respectively. In particular, if $\mathcal{C}$ is in $\mathbf{NC}^1$, then so is $\tilde{\mathcal{C}}$. Further, assume the CS scheme has decomposable online-offline efficiency (see Definition 4.2), and in particular, we have algorithms $(\mathsf{CS.Aggrgt}, \mathsf{CS.VrfyOnL})$.

Our construction of offline-online efficient HomSig with message space $\{0,1\}^\ell$ for circuit class $\mathcal{C}$ is provided as follows:

$\mathsf{HS.KeyGen}(1^\kappa, 1^\ell)$: On input the security parameter 1^κ and the message length 1^ℓ, generate $(\mathsf{msk}, \mathsf{vk}) \stackrel{\$}{\leftarrow} \mathsf{CS.Setup}(1^\kappa, 1^\ell)$, and output $\mathsf{pk} := \mathsf{vk}$ and $\mathsf{sk} := \mathsf{msk}$.

$\mathsf{HS.Sign}(\mathsf{sk}, C)$: On input $\mathsf{sk} = \mathsf{msk}$ and $C \in \mathcal{C}$, compute the associated circuit $\tilde{C} \in \tilde{\mathcal{C}}$ defined above. I.e., $\tilde{C}(\mathbf{x}', z') = \big(C(\mathbf{x}') \stackrel{?}{=} z'\big)$ for all $(\mathbf{x}', z') \in \{0,1\}^\ell \times \{0,1\}$. Then compute $\mathsf{sk}_{\tilde{C}} \stackrel{\$}{\leftarrow} \mathsf{CS.KeyGen}(\mathsf{msk}, \tilde{C})$, and output $\boldsymbol{\sigma} = \mathsf{sk}_{\tilde{C}}$.

$\mathsf{HS.Eval}(\mathsf{pk}, C, \mathbf{x}, \boldsymbol{\sigma})$: On input $C \in \mathcal{C}$, $\mathbf{x} \in \{0,1\}^\ell$, and $\boldsymbol{\sigma} = \mathsf{sk}_{\tilde{C}}$, compute $z = C(\mathbf{x})$. Then compute $\sigma_{\mathsf{CS}} \stackrel{\$}{\leftarrow} \mathsf{CS.Sign}(\mathsf{sk}_{\tilde{C}}, \mathbf{x} \| z)$, and output $\sigma = \sigma_{\mathsf{CS}}$. Here $\cdot\|\cdot$ denotes concatenation of bit strings.

$\mathsf{HS.Verify}(\mathsf{pk}, \mathbf{x}, z, \boldsymbol{\sigma})$: Run the following algorithms in order:

- $\mathsf{HS.VerifyOffL}(\mathsf{pk}, \mathbf{x})$: Parse $\mathsf{pk} = \mathsf{vk} \to (\mathsf{vk}_0, \{\mathsf{vk}_{i,b}\}_{i\in[n+1],b\in\{0,1\}})$ and compute $\mathsf{vk}_{\mathbf{x}} = \mathsf{CS.Aggrgt}(\{\mathsf{vk}_{i,\mathbf{x}_i}\}_{i\in[n]})$ where $\mathbf{x}_i$ denotes the i-th bit of $\mathbf{x}$. Finally, output $\mathsf{pk}_{\mathbf{x}} = (\mathsf{vk}_0, \mathsf{vk}_{\mathbf{x}}, \{\mathsf{vk}_{n+1,b}\}_{b\in\{0,1\}})$.
- $\mathsf{HS.VerifyOnL}(\mathsf{pk}', z, \sigma)$: Parse $\mathsf{pk}' \to (\mathsf{vk}_0, \mathsf{vk}_{\mathbf{x}}, \{\mathsf{vk}_{n+1,b}\}_{b\in\{0,1\}})$ and $\sigma \to \sigma_{\mathsf{CS}}$ and compute $\mathsf{vk}_{\mathbf{x}\|z} = \mathsf{CS.Aggrgt}(\mathsf{vk}_{\mathbf{x}}, \mathsf{vk}_{n+1,z})$. Then output whatever returned by running $\mathsf{CS.VrfyOnL}(\mathsf{vk}_0, \mathsf{vk}_{\mathbf{x}\|z}, \sigma_{\mathsf{CS}})$.

Security and correctness of the scheme directly follow from the result by Tsabary [Tsa17]. However, since we adapt slightly different syntax from hers, we show the proofs for the sake of completeness.

Correctness.

Theorem A.5 (Correctness). *If Π_{CS} satisfies correctness, then Π_{HS} satisfies correctness.*

Proof. It can be checked by inspection. Fix messages $\mathbf{x} \in \{0,1\}^\ell$ and a circuit $C \in \mathcal{C}$ and let $z = C(\mathbf{x})$, Then by correctness of Π_{CS}, we have $\mathsf{CS.Vrfy}(\mathsf{vk}, \mathbf{x}\|z, \mathsf{vk}_{\mathsf{CS}}) = \top$ where $\mathsf{vk}_{\mathsf{CS}} \xleftarrow{\$} \mathsf{CS.Sign}(\mathsf{sk}_{\tilde{C}}, \mathbf{x}\|z)$, since $\tilde{C}(\mathbf{x}\|z) = 1$ by definition of $\tilde{C}$. Therefore, due to the decomposable online-offline efficiency property of Π_{CS} (see Definition 4.2), it can be checked that $\mathsf{HS.Verify}$ output $\top$ as well. □

Security.

Theorem A.6 (Unforgeability). *If Π_{CS} satisfies (adaptive one-key) unforgeability, then Π_{HS} satisfies unforgeability.*

Proof. We prove by contradiction. Suppose that there exists a PPT adversary $\mathcal{A}$ that breaks unforgeability of Π_{HS} with non-negligible probability ϵ. We then construct a PPT adversary $\mathcal{B}$ that breaks the (adaptive one-key) security of Π_{CS} with probability ϵ as follows.

$\mathcal{B}(\mathsf{vk})$**:** It sets $\mathsf{pk} := \mathsf{vk}$ and gives the verification key pk to $\mathcal{A}$. When submits a circuit $C \in \mathcal{C}$, $\mathcal{B}$ first computes the associated circuit $\tilde{C} \in \tilde{\mathcal{C}}$ defined as $\tilde{C}(\mathbf{x}', z') = \left(C(\mathbf{x}') \stackrel{?}{=} z'\right)$ for all $(\mathbf{x}', z') \in \{0,1\}^\ell \times \{0,1\}$. $\mathcal{B}$ then makes a key generation query for $\tilde{C}$ to its own Π_{CS} challenger to obtain $\mathsf{sk}_{\tilde{C}}$, and gives $\mathsf{sk}_{\tilde{C}}$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(\mathbf{x}^\star, z^\star, \sigma^\star)$, $\mathcal{B}$ outputs $(\mathbf{x}^\star\|z^\star, \sigma^\star)$ as its forgery.

It is clear that $\mathcal{B}$ perfectly simulates the view of the unforgeability game of Π_{HS} to $\mathcal{A}$. We claim that whenever $\mathcal{A}$ wins the unforgeability game of Π_{HS}, $\mathcal{B}$ wins the unforgeability game of Π_{CS}. By the winning condition, $\mathcal{A}$'s forgery satisfies $(\mathbf{x}^\star, z^\star) \in \{0,1\}^\ell \times \{0,1\}$, $C(\mathbf{x}^\star) \neq z^\star$, and $\mathsf{HS.Verify}(\mathsf{pk}, \mathbf{x}^\star, z^\star, \sigma^\star) = \top$. In particular, this implies $\tilde{C}(\mathbf{x}^\star, z^\star) = 0$ and $\mathsf{CS.Vrfy}(\mathsf{vk}, \mathbf{x}^\star\|z^\star, \sigma^\star) = \top$. Therefore, $\mathcal{B}$ wins the unforgeability game of Π_{CS} with advantage ϵ, which is non-negligible as desired. □

Remark A.7. (Instantiations.) If we instantiate the scheme based on the decomposable online-offline efficient CS scheme in Section 4, we obtain a compact offline-online efficient HomSig scheme based on the DLIN assumption where "compact" means that the evaluated signature size does not depend on the size of the circuit on which the pre-evaluation signature is generated.

Remark A.8. (Extension to multi-data scheme) By using the generic transformation from single-data scheme to multi-data scheme by Gorbunov et al. [GVW15], we obtain multi-data HomSig scheme based on CS. Similarly to [KNYY19b], the resulting scheme *does not* satisfy amortized verification efficiency, which means that a verifier only needs to perform a computation depending on the size of the circuit only once and then he can reuse it for verifying signatures generated by multiple signers w.r.t. multiple data sets.

B Efficient Instantiation based on Groth-Sahai Proof.

Here, we give an efficient instantiation of the scheme given in Section 5.1 based on the Groth-Sahai proof under the SXDH assumption.

Groth-Sahai Proof. First, we briefly recall the Groth-Sahai proof. We refer the full description to [GS12]. We note that we focus on the instantiation based on the SXDH assumption here.

Roughly speaking, the Groth-Sahai proof is an efficient NIZK for a specific class of languages associated with a pairing-group. Specifically, it enables one to prove that committed values under an associated extractable[13] commitment scheme satisfy certain algebraic relations. We denote the setup algorithm of the associated commitment scheme by $\mathsf{GS.Setup}$ and committing algorithm in G_b by $\mathsf{GS.Commit}_{G_b}$ for $b = 1, 2$. The setup algorithm $\mathsf{GS.Setup}$ takes a description of a pairing group and outputs a common reference string $\mathsf{crs}_{\mathsf{GS}}$. The committing algorithm $\mathsf{GS.Commit}_{G_b}$ in G_b takes the common reference string $\mathsf{crs}_{\mathsf{GS}}$ and an element of G_b or $\mathbb{Z}_p$ as input, and returns a commitment that consists of 2 elements G_b. For $X = (X_1, ..., X_m) \in G_b^m$ (resp. $x = (x_1, ..., x_m) \in \mathbb{Z}_p^m$), we denote $\mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, X)$ (resp. $\mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, x)$) to mean $(\mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, X_1), ..., \mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, X_m))$ (resp. $(\mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, x_1), ..., \mathsf{GS.Commit}_{G_b}(\mathsf{crs}_{\mathsf{GS}}, x_m))$)for notational simplicity.

[13] Actually, the GS proof is a dual-mode NIZK, and satisfies perfect extractability or perfect zero-knowledge depending on the mode. Here, we focus on the binding mode where the perfect extractability is satisfied.

Suppose that $(X_1, ..., X_m) \in G_1^m$ and $(x_1, ..., x_{m'}) \in \mathbb{Z}_p^{m'}$ are committed under $\mathsf{GS.Commit}_{G_1}$ and $(Y_1, ..., Y_n) \in G_2^n$ and $(y_1, ..., y_{n'}) \in \mathbb{Z}_p^{n'}$ are committed under $\mathsf{GS.Commit}_{G_2}$. Then the GS proof enables one to prove an equation of the following forms.

Pairing product equation with target $1 \in G_T$:

$$\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m} \prod_{j=1}^{n} e(X_i, Y_j)^{\gamma_{ij}} = 1,$$

for constants $A_i \in G_1, B_i \in G_2, \gamma_{ij} \in \mathbb{Z}_p$.

Multi-scalar multiplication equation in G_1:

$$\prod_{i=1}^{n'} A_i^{y_i} \cdot \prod_{i=1}^{m} X_i^{b_i} \cdot \prod_{i=1}^{m} \prod_{j=1}^{n'} X_i^{\gamma_{ij} y_j} = T_1$$

for constants $A_i, T_1 \in G_1, b_i, \gamma_{ij} \in \mathbb{Z}_p$.

Multi-scalar multiplication equation in G_2:

$$\prod_{i=1}^{n} Y_i^{a_i} \cdot \prod_{i=1}^{m'} B_i^{x_i} \cdot \prod_{i=1}^{m'} \prod_{j=1}^{n} Y_j^{\gamma_{ij} x_i} = T_2$$

for constants $B_i, T_2 \in G_2, a_i, \gamma_{ij} \in \mathbb{Z}_p$.

Quadratic equation in $\mathbb{Z}_p$:

$$\sum_{i=1}^{n'} a_i y_i + \sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \gamma_{ij} x_i y_j = t \mod p$$

for constants $a_i, b_i, \gamma_{ij}, t \in \mathbb{Z}_p$.

We summarize the number of group elements that are needed to prove these equations in Table 3.

Table 3: Summary of Sizes of GS Proofs.

	G_1	G_2
Pairing product equation with target $1 \in G_T$	4	4
Multi-scalar multiplication equation in G_1	2	4
Multi-scalar multiplication equation in G_2	4	2
Quadratic equations in $\mathbb{Z}_p$	2	2

Our NIZK. 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 an $\mathbf{NC}^1$ 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 $\Pi_{\mathsf{SKE}} = (\mathsf{SKE.Setup}, \mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be an SKE scheme with message space $\{0,1\}^m$, key space $\{0,1\}^\ell$ and ciphertext space $\{0,1\}^{|\mathsf{ct}|}$. We require that it is one-time secure, its decryption circuit can be computed in $\mathbf{NC}^1$, it has no ciphertext overhead (i.e., $|\mathsf{ct}| = m$), and ℓ does not depend on m. We define a circuit $f_{\mathsf{pp}_{\mathsf{SKE}}} : \{0,1\}^{n'} \to \{0,1\}$ that computes

$$f_{\mathsf{pp}_{\mathsf{SKE}}}(K, x, \mathsf{ct}) = C(x, \mathsf{SKE.Dec}(\mathsf{pp}_{\mathsf{SKE}}, K, \mathsf{ct}))$$

for $\mathsf{pp}_{\mathsf{SKE}} \in \mathsf{SKE.Setup}(1^\kappa)$ where $n' := \ell + n + |\mathsf{ct}|$. Since we assume that both C and $\mathsf{SKE.Dec}$ can be computed in $\mathbf{NC}^1$, $f_{\mathsf{pp}_{\mathsf{SKE}}}$ can be computed in $\mathbf{NC}^1$. Let share be as defined in Definition 3.1 and EncInp and EncCir be as defined in Lemma 2.9.

Our scheme is described as follows.

$\mathsf{Setup}(1^\kappa)$**:**

1. Generate $\mathsf{pp}_{\mathsf{SKE}} \xleftarrow{\$} \mathsf{SKE.Setup}(1^\kappa)$.
2. Run $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{\$} \mathsf{GGen}(1^\kappa)$. and generate $\mathsf{crs}_{\mathsf{GS}} \xleftarrow{\$} \mathsf{GS.Setup}(\mathbb{G})$.
3. Sample $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{w}_i \xleftarrow{\$} \mathbb{Z}_p^2$ for $i \in [2n']$ and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^2$ and set
$$\mathsf{vk} = ([\mathbf{a}^\top]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, \ldots, [\mathbf{a}^\top \mathbf{w}_{2n'}]_1, e([\mathbf{a}^\top]_1, [\mathbf{v}]_2)).$$
4. Compute $\mathsf{EncCir}(f_{\mathsf{pp}_{\mathsf{SKE}}}) \to f'$, sample $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f', \mathbf{v})$ and $r_j \xleftarrow{\$} \mathbb{Z}_p$ for $j \in [\hat{m}]$ and compute
$$\mathsf{sk}_f = \left(\left\{ \mathsf{sk}_j := [r_j]_2, \quad \mathsf{sk}_{\rho(j),j} := [\mathbf{v}_j + \mathbf{w}_{\rho(j)} r_j]_2, \quad \{\, \mathsf{sk}_{i,j} := [\mathbf{w}_i r_j]_2 \,\}_{i\in[2n']\setminus\{\rho(j)\}} \right\}_{j\in[\hat{m}]} \right)$$
where $\mathbf{w}_0 = \mathbf{0}$ and $\hat{m}$ is the number of shares that is generated by $\mathsf{share}(f', \mathbf{v})$.
5. Output $\mathsf{crs} = (\mathsf{pp}_{\mathsf{SKE}}, \mathbb{G}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{vk}, \mathsf{sk}_f)$.

$\mathsf{Prove}(\mathsf{crs}, x, w)$**:**

1. Abort if $\mathcal{R}(x, w) = 0$. Otherwise, do the following.
2. Parse $\mathsf{crs} \to (\mathsf{pp}_{\mathsf{SKE}}, \mathbb{G}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{vk}, \mathsf{sk}_f)$.
3. Generate $K \xleftarrow{\$} \mathsf{SKE.KeyGen}(\mathsf{pp}_{\mathsf{SKE}})$ and $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{pp}_{\mathsf{SKE}}, K, w)$.
4. Set $z := \mathsf{EncInp}(K \| x \| \mathsf{ct})$ and compute ω_j such that $\mathbf{v} = \sum_{j:\rho(j)=0 \vee y_{\rho(j)}=1} \omega_j \mathbf{v}_j$ and
$$\sigma = \left(\sigma_1 = \prod_{j:\rho(j)=0 \vee z_{\rho(j)}=1} \left(\prod_{i:z_i=1} \mathsf{sk}_{i,j} \right)^{\omega_j}, \quad \sigma_2 = \prod_{j:\rho(j)=0 \vee z_{\rho(j)}=1} \mathsf{sk}_j^{\omega_j} \right) \in G_2^3.$$
5. Compute $Y := \prod_{i\in[n+|\mathsf{ct}|]} [\mathbf{a}^\top \mathbf{w}_{2\ell+2i-y_i}]_1$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$.
6. Compute $\mathsf{com}_K \xleftarrow{\$} \mathsf{GS.Commit}_{G_1}(\mathsf{crs}_{\mathsf{GS}}, \{K_i\}_{i\in[\ell]})$ and $\mathsf{com}_{\widetilde{K}_i} \xleftarrow{\$} \mathsf{GS.Commit}_{G_2}(\mathsf{crs}_{\mathsf{GS}}, \{\widetilde{K}_i\}_{i\in[\ell]})$ where $\widetilde{K} := K$.
7. Compute $\mathsf{com}_Z \xleftarrow{\$} \mathsf{GS.Commit}_{G_1}(\mathsf{crs}_{\mathsf{GS}}, Z)$ where $Z := \prod_{i\in[\ell]} [\mathbf{a}^\top \mathbf{w}_{2i-K_i}]_1 \cdot Y \in G_1$.
8. Compute $\mathsf{com}_V \xleftarrow{\$} \mathsf{GS.Commit}_{G_2}(\mathsf{crs}_{\mathsf{GS}}, V)$ where $V := [\mathbf{v}]_2 \in G_2^2$.
9. Compute $\mathsf{com}_\sigma \xleftarrow{\$} \mathsf{GS.Commit}_{G_2}(\mathsf{crs}_{\mathsf{GS}}, \sigma)$.
10. Generate a proof π_K that proves
$$K_i - \tilde{K}_i = 0 \mod p,$$
$$K_i \cdot \tilde{K}_i - K_i = 0 \mod p$$
for all $i \in [\ell]$. (This especially ensures $K_i, \widetilde{K}_i \in \{0,1\}$ for $i \in [\ell]$.)

11. Generate a proof π_Z that proves

$$Z = \prod_{i\in[\ell]} ([\mathbf{a}^\top \mathbf{w}_{2i-1}]_1^{\widetilde{K}_i} \cdot [\mathbf{a}^\top \mathbf{w}_{2i}]_1^{1-\widetilde{K}_i}) \cdot Y.$$

Note that this is equivalent to

$$Z = \prod_{i\in[\ell]} [\mathbf{a}^\top \mathbf{w}_{2i-\widetilde{K}_i}]_1 \cdot Y$$

for $\widetilde{K}_i \in \{0,1\}$.

12. Generate a proof π_V that proves

$$V = [\mathbf{v}]_2.$$

13. Generate a proof π_σ that proves

$$e([\mathbf{a}^\top]_1, \sigma_1) \cdot e(Z, \sigma_2)^{-1} \cdot e([\mathbf{a}^\top]_1, V)^{-1} = 1$$

14. Output $\pi := (\mathsf{ct}, \mathsf{com}_K, \mathsf{com}_{\widetilde{K}}, \mathsf{com}_Z, \mathsf{com}_V, \mathsf{com}_\sigma, \pi_K, \pi_Z, \pi_V, \pi_\sigma)$.

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$**:**

1. Parse $\pi \to (\mathsf{ct}, \mathsf{com}_K, \mathsf{com}_{\widetilde{K}}, \mathsf{com}_Z, \mathsf{com}_V, \mathsf{com}_\sigma, \pi_K, \pi_Z, \pi_V, \pi_\sigma)$. If it is not in this form, reject it. Otherwise, do the following.
2. Parse $\mathsf{crs} \to (\mathsf{pp}_{\mathsf{SKE}}, \mathbb{G}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{vk}, \mathsf{sk}_f)$.
3. Compute $Y := \prod_{i\in[n+|\mathsf{ct}|]} [\mathbf{a}^\top \mathbf{w}_{2\ell+2i-y_i}]_1$ where $y := (x, \mathsf{ct}) \in \{0,1\}^{n+|\mathsf{ct}|}$.
4. Output $\top$ if all proofs $(\pi_K, \pi_Z, \pi_V, \pi_\sigma)$ are valid w.r.t. commitments $(\mathsf{com}_K, \mathsf{com}_{\widetilde{K}}, \mathsf{com}_Z, \mathsf{com}_V, \mathsf{com}_\sigma)$ and otherwise $\bot$.

Efficiency. We summarize the size of each component of the proof in Table 4. The total proof size is $|w| + (6\ell + 14)|G_1| + (6\ell + 25)|G_2|$. Recall that ℓ is the length of the secret key of SKE and thus we can take ℓ to be fixed polynomial in the security parameter. In particular, under reasonable assumptions, we can take $\ell = O(\kappa)$.

Table 4: Summary of Sizes of GS Commitments and GS Proofs in Our Scheme.

	G_1	G_2
com_K	2ℓ	
$\mathsf{com}_{\widetilde{K}}$		2ℓ
com_Z	2	
com_V		4
com_σ		6
π_K	4ℓ	4ℓ
π_Z	4	2
π_V	4	8
π_σ	4	4
Total	$6\ell + 14$	$6\ell + 24$

Security. Since the above scheme is an instantiation of the scheme in Section 5.1, the security follows from Theorem 5.1 and Theorem 5.2. Specifically, we have the following theorems.

Theorem B.1 (Soundness). *The above NIZK scheme* Π_{NIZK} *is computationally (adaptive) sound if the SXDH assumption holds.*

Theorem B.2 (Zero-Knowledge). *The above NIZK scheme* Π_{NIZK} *is computationally zero-knowledge if the SXDH assumption holds and* Π_{SKE} *is one-time secure.*

Contents