Appeared in TCC-2010.

Efficiency Preserving Transformations for Concurrent Non-Malleable Zero Knowledge

Rafail Ostrovsky[1⋆], Omkant Pandey[1⋆⋆], and Ivan Visconti[2⋆⋆⋆]

[1] University of California, Los Angeles, USA, {rafail,omkant}@cs.ucla.edu
[2] University of Salerno, ITALY, visconti@dia.unisa.it

Abstract. Ever since the invention of Zero-Knowledge by Goldwasser, Micali, and Rackoff [1], Zero-Knowledge has become a central building block in cryptography - with numerous applications, ranging from electronic cash to digital signatures. The properties of Zero-Knowledge range from the most simple (and not particularly useful in practice) requirements, such as honest-verifier zero-knowledge to the most demanding (and most useful in applications) such as non-malleable and concurrent zero-knowledge. In this paper, we study the complexity of *efficient* zero-knowledge reductions, from the first type to the second type. More precisely, under a standard complexity assumption (DDH), on input a public-coin honest-verifier statistical zero knowledge argument of knowledge π' for a language L we show a compiler that produces an argument system π for L that is concurrent non-malleable zero-knowledge (under non-adaptive inputs – which is the best one can hope to achieve [2, 3]). If κ is the security parameter, the overhead of our compiler is as follows:

– The round complexity of π is $r + \tilde{O}(\log \kappa)$ rounds, where r is the round complexity of π'.
– The new prover $\mathcal{P}$ (resp., the new verifier $\mathcal{V}$) incurs an additional overhead of (at most) $r + \kappa \cdot \tilde{O}(\log^2 \kappa)$ modular exponentiations. If tags of length $\tilde{O}(\log \kappa)$ are provided, the overhead is only $r + \tilde{O}(\log^2 \kappa)$ modular exponentiations.

The only previous concurrent non-malleable zero-knowledge (under non-adaptive inputs) was achieved by Barak, Prabhakaran and Sahai [4]. Their construction, however, mainly focuses on a *feasibility* result rather than efficiency, and requires expensive $\mathcal{NP}$-reductions.

⋆ Supported in part by IBM Faculty Award, Xerox Innovation Group Award, the Okawa Foundation Award, Intel, Teradata, NSF grants 0716835, 0716389, 0830803, 0916574 and U.C. MICRO grant.

⋆⋆ Supported in part by NSF grants 0716835, 0716389, 0830803, 0916574, and the European Commission grants of the third author

⋆⋆⋆ Supported in part by the European Commission through the EU IST program under Contract IST-2002-507932 ECRYPT, and through the EU ICT program under Contract ICT-2007-216646 ECRYPT II.

1 Introduction

In this paper, we consider Zero-Knowledge argument systems that are non-malleable and secure against concurrent man-in-the-middle attacks. In such systems, the adversary has complete control over the communication channel and can behave as honest prover and honest verifier in any polynomial number of protocols, therefore controlling the scheduling of the messages. We aim at designing efficient argument systems secure against these attacks, namely *efficient concurrent non-malleable zero knowledge argument systems*. Despite the extreme importance of these proof systems, no efficient and secure (plain model) protocol for such settings is known until today. Feasibility results have been given originally by Dolev, Dwork, and Naor (DDN) [5], restricting the adversary to two simultaneous proofs. In recent work, Barak, Prabhakaran and Sahai [4] have obtained concurrent and non-malleable zero-knowledge without restricting the adversary to a bounded number of proofs, however the solutions proposed there can be viewed as constructing feasibility results only as their methods require NP reductions and are highly inefficient.

The need of *efficient* instantiations of concurrent NMZK for useful languages and its applicability as sub-protocols motivated the introduction of several strong set-up assumptions [6–8]. In this paper, we focus on achieving efficient transformations in the *plain* model which does not rely on any setup assumptions. We show a transformation that on input a public-coin honest-verifier statistical zero knowledge argument of knowledge π' for a language L produces a *concurrent non-malleable* zero-knowledge argument system π for L. Further, our transformation is an *efficiency preserving transformation* that does not require any $\mathcal{NP}$-reduction and works assuming standard number-theoretic assumptions (see theorem 1 for a precise statement).

It should be noted that CNMZK arguments are significantly harder to construct and analyze. In fact, Lindell proved that in the most general form of the attack, (non-trivial) CNMZK arguments do not even exist [2, 3]. However, assuming that the honest parties' inputs are fixed in advance (i.e., are not chosen *adaptively* based on the protocol execution), CNMZK was shown to be achievable by Barak, Prabhakaran, and Sahai (BPS). The impossibility results discussed in [2, 3, 9] and the plausibility results of [4] suggest that CNMZK (under the non-adaptive input notion) is the best notion of security for proof systems that one can hope to achieve in the plain model. Our results are the *first* efficiency preserving transformation for the concurrent man-in-the-middle setting in the plain model, gaining dramatic efficiency improvements over [4] (see further discussion on efficiency immediately after the statement of our main result).

OUR RESULTS. Assuming the hardness of (standard) decisional Diffie-Hellman assumption, we show CNMZK argument-of-knowledge (see theorem 1). Our results require that the HVZK argument system admit *statistical* simulation and be an "argument of knowledge"[3]. We remark that the statistical simulation requirement for the given HVZK argument, is easy to achieve as most HVZK protocols that we know of already admit statistical simulation (by using statistically hiding commitments such as [10] – which exist under the DDH assumption).

Theorem 1 (Main Result). *Let $\pi' : \langle \mathcal{P}^{'}, \mathcal{V}^{'} \rangle$ be a* public coin *honest verifier* statistical *zero-knowledge argument of knowledge, for some language $L \in \mathcal{NP}$. Let κ be a security parameter, and q be a prime number whose length is determined by κ. Then, assuming that the Decisional Diffie-Hellman Assumption holds, it is possible to transform π' into a new argument system $\pi : \langle \mathcal{P}, \mathcal{V} \rangle$ such that,*

- *Protocol π is a* computational concurrent non-malleable zero-knowledge *argument of knowledge for L.*
- *Protocol π has $r + \tilde{O}(\log \kappa)$ rounds of interaction, where r is the round complexity of π'.*
- *The new prover $\mathcal{P}$ (resp., the new verifier $\mathcal{V}$) incurs an additional overhead of $r + \kappa \cdot \tilde{O}(\log^2 \kappa)$ exponentiations in $\mathbb{Z}_q$. For tag-based non-malleability, the overhead is only $r + \tilde{O}(\log^2 \kappa)$ additional exponentiations in $\mathbb{Z}_q$, assuming tags of length $\tilde{O}(\log \kappa)$.*

Although our main focus is the plain model, our results about tag-based non-malleability, lead to more efficient constructions in the Bare-Public-Key (BPK) model [11]. The BPK model, assumes an *untrusted* setup which brings it very close to the plain model. Like the plain model, our results in the BPK model are the first efficient transformations (see section 5 for more details).

Our starting point to avoid $\mathcal{NP}$-reductions is "Simulatable Commitments" as defined by Micciancio and Petrank [12] (though our construction and proof requires development of several new techniques and ideas on top of this work). Using simulatable commitments, Micciancio and Petrank demonstrate how to efficiently transform any HVZK argument system into a *concurrent* ZK argument system which is secure against a cheating verifier V^* mounting a *concurrent* attack. Their transformation increases the round complexity of the original argument system by $\tilde{O}(\log \kappa)$ and incurs an additional overhead of $r + \tilde{O}(\log \kappa)$ exponentiations in $\mathbb{Z}_q$.

TECHNICAL OVERVIEW AND MAIN DIFFICULTIES. We design a new protocol to make the given protocol π_{HV} secure in the CNMZK model without much compromise in its efficiency. As we explain below, our transformation is conceptually different from the only known CNMZK protocol of BPS. Due to this conceptual difference in the construction, our proof of security is entirely new.

[3] The "argument of knowledge" requirement is actually due to the particular definition of security we aim to achieve, namely *simulation extractability* (see Definition 1). If the given protocol is not an argument of knowledge, our transformation still delivers *simulation soundness.*

To explain the main conceptual ideas/differences, we now sketch our transformation.[4] At a very high level, our transformation has following structure: (1) Our verifier, $\mathcal{V}$, first executes a KP/PRS preamble for a secret v; (2) Our prover, $\mathcal{P}$, then commits to 0^κ using a (properly instantiated) DDN-commitment; (3) $\mathcal{V}$ then reveals v; (4) and finally, $\mathcal{P}$ proves to $\mathcal{V}$ that "$x \in L$ OR $\mathcal{P}$ committed to v".

Note that in phase-(3) we need an *efficient* version of DDN-commitments.[5] Also, phase 4 needs typically required NP reductions to apply "FLS"-trick which requires an NP reduction. Instead, we design a new protocol by extending and applying in a non-trivial way the Micciancio-Petrank (MP) [12] transformation to the input protocol π_{HV}. We now explain the main conceptual differences from BPS and their proof.

Note that our protocol has only four phases whereas BPS has five: we do not require a separate phase involving a statistically hiding commitment to 0^κ, followed by a SZKAOK for the knowledge of randomness to the commitment. This phase is *crucial* for BPS-proof to go through. This changes the proof significantly – we directly rely on phase-(3) and phase-(4). Next, it is clear that simulation will proceed by $\mathcal{S}$ extracting v (from KP/PRS-preamble) and then committing to v (instead of 0^κ) in the left sessions of DDN. Protocol of BPS commits to the *witness* (of the statement) instead and relies on this stage for extraction. Clearly, this completely changes how our extractor would work. Instead, we must rely on the last phase to perform extraction. This is more involved than it seems: when simulator uses commitment to v as witness for succeeding in the last phase. Thus, to be able to argue correctness of extraction, we need *statistical simulation*.[6] Unfortunately, because of MP-transformation, transformed π_{HV} loses its statistical simulation – making the proof stuck. However, we identify a new property: MP-transformation admits "statistical simulation with respect to *lucky* provers" (see section 3), and this suffices to argue the correctness of extraction. Briefly. a "lucky" prover is one who can *guess* the PRS-secret correctly, in advance. Our extractor also differs from "standard" methods: we first test whether man-in-the-middle has succeeded in setting up a trapdoor by doing a preliminary DDN-extraction before performing actual extraction from the last phase (otherwise the extractor may not be expected-PPT).

Other Related Work. Achieving practical constructions/instantiations of advanced cryptographic tasks has become an increasingly popular research direction in recent years. To gain efficiency, $\mathcal{NP}$-reductions has been a common bot-

---

[4] Unfortunately, here reader's familiarity with the BPS-protocol and their proof structure is required.

[5] Interestingly, this is not immediately clear. Before this work, to the best of our knowledge, the only hope for achieving an efficient non-malleable commitment was from a *recent* protocol of Lin, Pass, and Venkitasubramaniam [13]. Here, we show a new and simple technique which provides an efficient instantiation of DDN-commitments (see section 3.2).

[6] This is a somewhat common issue in non-malleability proofs when going from one hybrid to another (e.g., the non-malleable commitments of Pass and Rosen [14]).

tleneck that most of these research works also aim at avoiding. Among these, the most relevant works are those of Garay, MacKenzie, and Yang [6], and De Santis, Di Crescenzo, Ostrovsky, Persiano, and Sahai [15] (CRS model), and Micciancio and Petrank [12] (plain model). In the area of secure two-party computation, see the works of Mohassel and Franklin [16], Woodruff [17], Lindell and Pinkas [18], and Goyal, Mohassel, and Smith [19]. For non-interactive zero-knowledge see Chase and Lysyanskaya [20], and Groth, Ostrovsky, and Sahai [21].

2 Definitions

In this section we present relevant definitions. We assume familiarity with (standard) cryptographic concepts such as computational and statistical indistinguishability, $\mathcal{NP}$-relations, interactive proof and argument systems, simulation paradigm, etcetera (see [22]). In the following, L is an $\mathcal{NP}$-language with witness relation R_L. That is, a statement $x \in L$ iff there exists a y of length poly$(|x|)$ such that $R_L(x, y) = 1$.

Concurrent Man-in-the-Middle Attack. The concurrent man-in-the-middle setting proceeds as follows. First, the inputs to the honest provers, i.e., statements $x_1, \ldots, x_{m_L} \in L \cap \{0,1\}^n$ are chosen; thereafter, m_L honest provers, $P_i \overset{\text{def}}{=} P(x_i, y_i; \omega_i)$, are constructed (for $i \in [m_L]$) such that $R_L(x_i, y_i) = 1$, and ω_i is a uniformly chosen random tape of sufficient (polynomial in κ) length. Adversary M may now start interacting with these provers while playing the role of a verifier of π with each one of them. These interactions are called "left" interactions. At any point, M, may adaptively output a new statement $\tilde{x}_i \in L \cap \{0,1\}^n$. Whenever it does so, an honest verifier $V_i \overset{\text{def}}{=} V(\tilde{x}_i; \tilde{\omega}_i)$, is created with input $\tilde{x}_i$ and uniformly chosen randomness $\tilde{\omega}_i$. Such verifiers are created to the "right" of M who may try to convince V_i of the validity of statement $\tilde{x}_i$ by playing the role of the prover in a session of π. These interactions are called the "right" interactions, and M may simultaneously continue its left interactions. Let m_R denote the number of right hand side sessions before M halts.

A *concurrent non-malleable attack*, a man-in-the-middle adversary M interacts with provers $P_1, \ldots, P_{m_L}$ in m_L "left sessions" and verifiers $V_1, \ldots, V_{m_R}$ in m_R "right sessions" of the protocol with M controlling the scheduling of all the sessions. "Left inputs" $x_1, \ldots, x_{m_L}$ are fixed in advance, whereas "right inputs" $\tilde{x}_1, \ldots, \tilde{x}_{m_R}$ can be decided by M adaptively. We consider only non-uniform PPT adversaries M, and so both m_L, m_R are polynomial in κ.

Following the work of Pass and Rosen [14], when dealing with non-malleability it is sometimes easier to work with a somewhat stronger notion called the *simulation-extractability*. They demonstrate that simulation-extractability implies non-malleable zero-knowledge argument (proof) of knowledge property. This approach was also followed by BPS, and we stick to their definition.

Definition 1. *A protocol* $\pi \stackrel{def}{=} \langle P, V\rangle$ *is said to be a Concurrent Non-Malleable Zero Knowledge (*CNMZK*) argument of knowledge for membership in an* $\mathcal{NP}$ *language* L *with witness relation* R_L*, if it is an interactive argument system between a prover and verifier (both PPT) such that the following conditions hold.*

Completeness *For every* x, y *such that* $R_L(x, y) = 1$*,* $P(x, y)$ *makes* V *accept with probability 1.*

Soundness, Zero Knowledge, and Non-malleability *For every PPT adversary* M *launching a concurrent non-malleable attack as above (i.e.,* M *interacts with* $P_1, \ldots, P_{m_L}$ *in "left sessions" and* $V_1, \ldots, V_{m_R}$ *in right sessions as defined above), there exists an expected polynomial time simulator-extractor* $\mathcal{S}$ *such that for every set of "left inputs"* $x_1, \ldots, x_{m_L}$ *we have* $\mathcal{S}(x_1, \ldots, x_{m_L}) = (\nu, \tilde{y}_1, \ldots, \tilde{y}_{m_R})$ *such that,*

- ν *is the simulated joint view of* M *and* $V_1, \ldots, V_{m_R}$*. Further, for any set of witnesses* $(y_1, \ldots, y_{m_L})$ *defining the provers* $P_1, \ldots, P_{m_L}$*, the view* ν *is distributed computationally indistinguishably from the view of* M *in a real execution.*
- *In the view* ν*, let* $\textsc{trans}_h$ *denote the transcript of* h^{th} *left execution, and* $\tilde{\textsc{trans}}_\ell$ *that of* ℓ^{th} *right execution,* $h \in [m_L], \ell \in [m_R]$*. If* $\tilde{x}_\ell$ *is the common input in* $\tilde{\textsc{trans}}_\ell$*,* $\tilde{\textsc{trans}}_\ell \neq \textsc{trans}_h$ *(for all* h*) and* V_ℓ *accepts, then* $R_L(\tilde{x}_\ell, \tilde{y}_\ell) = 1$ *except with probability negligible in* κ*.*

The probability is taken over the random coins of $\mathcal{S}$*. Further, the protocol is* black-box CNMZK*, if* $\mathcal{S}$ *is an universal simulator that uses* M *only as an oracle, i.e.,* $\mathcal{S} = \mathcal{S}^M$*.*

The second condition in the definition of soundness above, says that if some right session is not an *exact copy* of any of the left sessions, then $\mathcal{S}$ should output a valid witness for the statement of that right session.

The DDH Assumption. Let q be a sufficiently large randomly chosen prime such that there exists another sufficiently large prime p that divides $q-1$. Let G_p be an order p (multiplicative cyclic) subgroup of $\mathbb{Z}_q$, with some generator g. Then, the DDH assumption states that for randomly and independently chosen $a, b, c \in \mathbb{Z}_p$, the following two distributions are computationally indistinguishable: $\left(g^a, g^b, g^{ab}\right)$ and $\left(g^a, g^b, g^c\right)$.

Strong Signatures. A signature scheme $(\mathcal{K}, \textsc{sign}, \textsc{verify})$ is said to be *strongly unforgeable* if no efficient adversary, with access to a signing oracle with respect to verification key VK, can output a pair (m, σ) with non-negligible probability, such that: $\textsc{verify}(m, \sigma, \textsc{vk}) = 1$ and the pair (m, σ) does not correspond to the input-output pair of a performed oracle query. A *strong signature scheme* is a signature scheme that is strongly unforgeable.

Notation. Throughout the paper, $\mu : \mathbb{N} \to \mathbb{R}$ denotes a negligible function in κ (the security parameter). If a message u appears in the "left" session, then its counterpart in the "right" session will be denoted by $\tilde{u}$.

3 Building Blocks: Efficient Instantiations

We discuss two of our main building blocks: **(a)** simulatable commitments, and **(b)** DDN-commitments. We assume here familiarity with *commitment schemes* and their computational/statistical/perfect binding and hiding properties (see [22]). Simulatable commitments were used by [12] to compile HVZK arguments into concurrent ZK arguments (for us just stand-alone ZK suffices) – which we discuss briefly. Thereafter, we discuss an efficient implementation of the DDN-commitment scheme. Our descriptions are brief, and we refer the reader to the respective works for more details.

3.1 Simulatable Commitments

A simulatable commitment [12] scheme is a tuple $(\text{COM}, \text{DCOM}, P_{\text{COM}}, V_{\text{COM}}, S_{\text{COM}})$ such that $(\text{COM}, \text{DCOM})$ specifies an usual (*non-interactive, perfectly binding, computationally hiding*) commitment scheme. Additionally, it comes with a *3-round* HVZK proof system $(P_{\text{COM}}, V_{\text{COM}})$ to show, given two strings (c, v), c is a commitment to v (i.e., $\exists r$ s.t. $c \leftarrow \text{COM}(v; r)$). The proof system has perfect completeness, optimal soundness[7], and efficient prover (given input r); S_{COM} is the simulator for the HVZK property of the system. A construction of a simulatable commitment scheme, based on the DDH assumption, is given in the full version of this paper (the construction is due to [12], and admits *statistical* simulation). Note that because of the (computational) hiding property, it follows that the output of S_{COM} on input a true statement (c, v), is computationally indistinguishable from its output on input a false statement (c, v').

HVZK to Stand-alone ZK. Using simulatable commitments, [12] show how to transform any public coin HVZK argument system $\pi_{\text{HV}} : \langle P_{\text{HV}}, V_{\text{HV}} \rangle$ to a new system, which is zero-knowledge with respect to *any* (PPT) verifier (i.e., the new system is stand-alone ZK). We call this transformation the Micciancio-Petrank transformation, and denote the new system by $\pi_{\text{MP}} : \langle P_{\text{MP}}, V_{\text{MP}} \rangle$.[8]

To pinpoint a crucial property we need, we briefly explain how the transformed protocol π_{MP} proceeds. First, parties P_{MP} and V_{MP} execute a preamble phase, in which V_{MP} commits to a value $v \in \mathbb{Z}_p$, using a statistically hiding commitment; P_{MP} then commits to 0^κ using *simulatable commitments* (let the commitment be denoted by c). Finally V_{MP} opens the value v to P_{MP}. The transcript of conversation is thus (c, v). Now the second phase of the proof starts, in which V_{MP} acts like V_{HV}, but each challenge of V_{MP} is decided using "coin-tossing"-type style (see the full version of this paper for concrete details). The proof system, that comes with simulatable commitments, is used for this purpose with input statement (c, v). Statement (c, v) is false in a real execution with

[7] Informally, it means that for a false statement (c, v), given the first prover-message and the verifier-query, there is exactly one convincing answer.

[8] As mentioned earlier, the main result of [12] gives *concurrent* ZK; stand-alone ZK is a special case and adds only *four* more rounds to π_{HV}.

high probability (which results in uniform output for V_{HV}'s challenges), but the simulator can setup a *true* (c, v) via rewinding (and hence bias the output of coin-tossing to any value). The protocol is thus both: ZK and sound. The crucial property that we need, is described next.

Statistical *Simulation with respect to "lucky" Provers.* In general, π_{MP} is only computational ZK, since the prover commits to 0^κ while the simulator commits to v (the message opened by V_{MP}). However, consider a prover who can always guess the value v correctly and commits to it instead of 0^κ (but uses its witness in the rest of the execution of P_{MP}). Call such a prover "lucky".[9] Then, for such provers, the statement (c, v) (from first phase) is always *true*. Thus, if π_{HV} admits *statistical* simulation, the protocol π_{MP} also admits statistical simulation *with respect to the "lucky" provers.* Formally, there exists a simulator S_{MP} for every verifier V^*_{MP} (of the protocol π_{MP}) such that the output of S_{MP} is *statistically indistinguishable* from the view of V^*_{MP} in a real execution with a "lucky" prover (say $P^{(\text{lucky})}_{\text{MP}}$).

3.2 The DDN Commitment

Our construction needs an efficient instantiation of the DDN-commitment protocol. The DDN-commitment protocol is *non-malleable* which means – intuitively – given a commitment c on some message v, knowledge of c does not help a man-in-the-middle adversary in constructing a new commitment c' of a *related* message v'. The formal definition that we shall stick to appears in the full version of this paper.(This definition is satisfied by the variant of DDN-commitment protocol given in [4].) An efficient instantiation appears in Fig. 1.

In step 2, we mention the use of an efficient SZKAOK. An appropriate SZKAOK would be the one obtained by *sequentially* repeating the Schnorr protocol [23] $\omega(1)$ times. The size of verifier's challenge in each execution of Schnorr protocol, however, would only be $\log \kappa$.

In step 3 of the BCK protocol, we need an efficient proof system for statements of type: "c, c_{1-r} are commitments to v, x_{1-r} resp., s.t. $\alpha = x_{1-r} + v \mod p$". Informally, it can be achieved as follows. Commitment c is a pair of values in $\mathbb{Z}_q$: (a, b). Similarly, $c_{1-r} = (a', b')$. Compute $A = aa' \mod q, B = bb' \mod q$. Now use the proof system of simulatable commitments, to prove that (A, B) is a commitment to α. (Note that the proof system is only HVZK, but it can be first converted to (general) ZK by using the Micciancio-Petrank transformation once again before it is used in step 3 of the BCK protocol). The details are an easy exercise, which we defer to the full version of the paper.

[9] Note that in real executions provers will not be "lucky" w.h.p.; the simulator will, however, setup the situation of the "lucky" prover to succeed.

The DDN-commitment protocol.

1. $\mathcal{S}_{\text{DDN}}$ sends VK – the verification key of a strong signature scheme, computed using $\mathcal{K}(1^\kappa)$. Let $|\text{VK}| = \kappa$.
2. $\mathcal{S}_{\text{DDN}}$ commits to v using the simulatable commitment scheme (COM, DCOM), and sends $c \leftarrow \text{COM}(v;\omega)$ to $\mathcal{R}_{\text{DDN}}$. $\mathcal{S}_{\text{DDN}}$ then proves to $\mathcal{R}_{\text{DDN}}$ the knowledge of (v,ω) using an efficient $\omega(1)$-round public-coin statistical ZK argument of knowledge (SZKAOK). The last message of this SZKAOK is called the "Knowledge Determining Message" (KDM).
3. For $i = 1,\ldots,\kappa$, define $t^{(i)} = i \circ \text{VK}_i$. Thus, $|t^{(i)}| = 1 + \log\kappa$. Let $\text{BCK}^{\|}$ denote the protocol obtained by composing β parallel executions of the BCK protocol (described below), here $\beta \in \omega(\log\kappa)$. Recall that DDN defines two types of scheduling for $\text{BCK}^{\|}$: type-0 and type-1 (see [5]).
4. In *parallel*, for $i = 1,\ldots,|\text{VK}|$, execute the following protocol
   – For $j = 1,\ldots,(1+\log\kappa)$ do *sequentially* –
     Execute $\text{BCK}^{\|}$ with type-$t_j^{(i)}$ scheduling.
     Execute $\text{BCK}^{\|}$ with type-$(1 - t_j^{(i)})$ scheduling.
5. $\mathcal{S}_{\text{DDN}}$ signs the full transcript of execution, and sends the signature σ to $\mathcal{R}_{\text{DDN}}$. $\mathcal{R}_{\text{DDN}}$ verifies the signature.

The BCK protocol mentioned in step 3 above.

1. $\mathcal{S}_{\text{DDN}}$ chooses $x_0, x_1 \in \mathbb{Z}_p$, and commits to each one of them using simulatable commitments; $c_b \leftarrow \text{COM}(x_b;\omega_b), b \in \{0,1\}$. (Step BCK1)
2. $\mathcal{R}_{\text{DDN}}$ sends a bit r to $\mathcal{S}_{\text{DDN}}$. (Step BCK2)
3. $\mathcal{S}_{\text{DDN}}$ opens x_r and sends $\alpha = x_{1-r} + v \mod p$. $\mathcal{S}_{\text{DDN}}$ then proves to $\mathcal{R}_{\text{DDN}}$ using an efficient ZK protocol that: "c, c_{1-r} are commitments to v, x_{1-r} resp., s.t. $\alpha = x_{1-r} + v \mod p$". This protocol is discussed in section 3.2. (Step BCK3)

Fig. 1. The $O(\log\kappa)$-round DDN commitment scheme. $\mathcal{S}_{\text{DDN}}$ holds a value $v \in \mathbb{Z}_p$.

4 An Efficiency Preserving Transformation

4.1 The Extraction Preamble

The extraction preamble is just the the "KP/PRS-preamble". This is a protocol between two players: a sender, A, and a receiver, B. The sender holds a value $v \in \mathbb{Z}_p$.[10] Let $a_i \stackrel{\text{def}}{=} \{(v_0^{i,j}, v_1^{i,j})\}_{j=1}^{\beta}$ be the list of pairs such that $v_0^{i,j} + v_1^{i,j} = v \mod p$ where values $v_b^{i,j} \in \mathbb{Z}_p$ for all $b \in \{0,1\}$ and $i,j \in [\beta]$. Here $\beta = \beta(\kappa)$ is any function in $\omega(\log\kappa)$. So there are β such lists, each consisting of β pairs.

[10] Here, and everywhere else in this paper, when we mention $\mathbb{Z}_p$, it should be assumed that $\mathbb{Z}_p$ is an appropriately chosen order p subgroup of $\mathbb{Z}_q$ in which DDH is hard, where p,q are as defined in the DDH assumption.

The preamble consists of three steps. First step is the *commitment* step. Sender A chooses the parameters for the (perfectly binding) simulatable commitment scheme[11], and sends commitments to value v and to each share $v_b^{i,j} \in \mathbb{Z}_p$ (defined as above), using COM. The second step, (called the *challenge-response* step), is an interactive protocol consisting of β rounds, where in round i, player B sends a *challenge* $r_i \in \{0,1\}^\beta$, and A sends a *response* as follows. The response of A consists of an opening of the commitments to one of the elements of each pair in a_i. That is, if $r_i^j = b$ (the j^{th}-bit of r_i), then A includes in its response the value $v_b^{i,j}$, and the randomness it used to commit to $v_b^{i,j}$. At the end of this step, we say that the preamble has *concluded*. The final step is the *opening* step. This step consists of A sending to B, the decommitment information corresponding to *all* the commitments of the *commitment* step. That is, A sends to B the values $v, v_b^{i,j}$, and the randomness it used to commit to them.

There can be other messages in the protocol between the prover concluding the preamble and the verifier opening the commitments. It is easy to see that if COM is a commitment scheme[12], the extraction-preamble is an interactive commitment scheme. We now state a result from PRS [25].

Lemma 1. *(Adapted from [25]) Consider provers $P_1, \ldots, P_m$ and an adversarial verifier $\mathcal{A}_{\text{PRS}}$ running m sessions of a protocol with the extraction-preamble as described above, where m is polynomial in κ. Then except with negligible probability in κ, in every thread of execution output by the KP/PRSsimulator, if the simulation reaches a point where P_i accepts the extraction-preamble with v as the secret of the sender (in that particular thread), then at the point when the preamble was concluded, the simulator would have already recorded the value v.*

In fact, we will also need a refinement of this lemma. However, both the lemma and the refinement are not needed until the analysis of hybrid simulators (which appears in the full version of this paper). Thus, the refinement and a more detailed discussion is provided in the full version of this paper.

4.2 The Transformation

Overview. We provide an overview of our transformation here in order to present the basic ideas in the construction (issues originating in the proof due to these ideas, were discussed in the introduction). The transformed protocol has the following structure. In the first phase, the verifier $\mathcal{V}$ executes the extraction preamble (of $\widetilde{O}(\log \kappa)$ rounds with a value $v \in \mathbb{Z}_p$ chosen uniformly. In the second phase, the prover commits to 0^κ using our efficient DDN-commitment

[11] These commitments will sometimes be referred to as Micciancio-Petrank commitments.

[12] In our description, COM is chosen to be a simulatable commitment which is perfectly binding. For the extraction-preamble, however, a perfectly hiding commitment scheme (such as [10]) may be used as well. Also, for simplicity, we have chosen to use the extraction preamble in the PRS-style, but the original style of Richardson-Kilian [24] will be more efficient.

scheme. Note that the first message of this DDN-commitment phase includes a perfectly binding commitment to 0^κ using a simulatable commitment scheme – which we denote by c^*. $\mathcal{V}$ now opens the value v in the preamble (along with opening all other commitments of the preamble). This defines the pair (c^*, v).

Let the input protocol be $\langle \pi_{\text{HV}} \rangle$. Recall that the Micciancio-Petrank transformation goes in two steps. In the first step a preamble is run, to obtain a pair (c_1, v_1) and then the second step uses this pair to enforce random challenges from the verifier of π_{HV}. In our protocol also, both $\mathcal{P}, \mathcal{V}$ now proceed exactly like this transformation, except that the first step of the transformation is not executed. Instead, (c^*, v) is used in place of (c_1, v_1).(We also use the standard trick of sending a verification key VK of a strong signature scheme to be used as the identity for the DDN-commitment, and in the end sign the whole transcript).

The formal description of our transformation (sometimes also referred to as the *compiler*) is given in Figure 2. The compiler transforms any given public coin statistical HVZK argument of knowledge in a CNMZK argument of knowledge.

The Compiler (from HVZK to CNMZK): The given HVZK argument is $\pi_{\text{HV}} : \langle P_{\text{HV}}, V_{\text{HV}} \rangle$.
$\mathcal{P} \rightarrow \mathcal{V}$: Run the key generation algorithm, $(\text{VK}, \text{SK}) \leftarrow \mathcal{K}(1^\kappa)$. Send VK to $\mathcal{V}$.

$\mathcal{V} \leftrightarrow \mathcal{P}$: $\mathcal{V}$ chooses a value $v \in_r \mathbb{Z}_p$. $\mathcal{V}$ and $\mathcal{P}$ then execute the "extraction preamble" where $\mathcal{V}$ plays the role of the sender, with input v; $\mathcal{P}$ plays the role of the receiver. Let $c \leftarrow \text{COM}(v; \omega), c_b^{i,j} \leftarrow \text{COM}(v_b^{i,j}; \omega_b^{i,j})$ denote the corresponding commitments. Recall that: $b \in \{0,1\}, i, j \in [\beta], v_0^{i,j} + v_1^{i,j} = v$. Here, $\omega, \omega_b^{i,j}$ denote the randomness used by the commitment scheme.

$\mathcal{P} \leftrightarrow \mathcal{V}$: $\mathcal{P}$ and $\mathcal{V}$ execute a DDN-commitment protocol in which $\mathcal{P}$ plays the role of $\mathcal{S}_{\text{DDN}}$ with input 0^κ, and $\mathcal{V}$ plays the role of $\mathcal{R}_{\text{DDN}}$. Let $\text{FM}^* = (p^*, q^*, g^*, h^*, \text{VK}^*, c^* \leftarrow \text{COM}(0^\kappa; \omega^*))$, denote the first message of the DDN-commitment protocol.

$\mathcal{V} \rightarrow \mathcal{P}$: $\mathcal{V}$ executes the *opening* step of the "extraction preamble", by sending the opening of all commitments sent in phase 2. That is, $\mathcal{V}$ sends to $\mathcal{P}$ the values: $v, v_b^{i,j}$ and randomness $\omega, \omega_b^{i,j}$, where b, i, j are defined as above.

$\mathcal{P} \leftrightarrow \mathcal{V}$: $\mathcal{P}$ (resp., $\mathcal{V}$) applies the Micciancio-Petrank transformation to P_{HV} (resp., V_{HV}) to obtain the algorithm P_{MP} (resp., V_{MP}). Now, $\mathcal{P}$ and $\mathcal{V}$ execute the $(P_{\text{MP}}, V_{\text{MP}})$ protocol with common input (x, c^*, v) in which $\mathcal{P}$ (using P_{MP}) proves to $\mathcal{V}$ (using V_{MP}) that $x \in L \cap \{0,1\}^n$.

$\mathcal{P} \rightarrow \mathcal{V}$: Let trans denote the transcript of communication so far. $\mathcal{P}$ computes $\sigma \leftarrow \text{SIGN}(\mathsf{trans}, \text{SK}, \text{VK})$, and sends σ to $\mathcal{V}$.

Fig. 2. The Transformed Argument System $\pi : \langle \mathcal{P}, \mathcal{V} \rangle$

In all steps above, whenever a message is not according to the protocol specifications, an honest party aborts the protocol.[13] We will frequently refer to above steps as *phases*. Thus, our transformation has six phases, where in phase-1 $\mathcal{P}$ sends a verification key to $\mathcal{V}$, in phase-2 $\mathcal{V}$ and $\mathcal{P}$ execute an extraction preamble, and so on.

4.3 Proving Concurrent Non-malleability

We now proceed to the actual proof that $\pi : \langle \mathcal{P}, \mathcal{V} \rangle$ is indeed a CNMZK argument of knowledge, given that $\pi_{\rm HV} : \langle P_{\rm HV}, V_{\rm HV} \rangle$ is a public coin honest verifier statistical zero-knowledge argument of knowledge. Using a series of hybrid simulators, we will show how to simulate the joint view of M and $V_1, \dots, V_{m_R}$, while simultaneously extracting a witness for each $\tilde{x}_\ell$ whenever V_ℓ's view is accepting and $\tilde{\text{TRANS}}_\ell \neq \text{TRANS}_h$ (for all h). Assume M to be deterministic, without loss of generality. It is easy to see that if $\tilde{\text{VK}}_\ell = \text{VK}_h$ for some ℓ, h (i.e., M copies the tag), then due to the *strong unforgeability* of the signature scheme, it holds that $\tilde{\text{TRANS}}_\ell = \text{TRANS}_h$ except with negligible probability. Thus, in the proof we will not attempt to extract a witness for $\tilde{x}_\ell$ whenever $\tilde{\text{VK}}_\ell = \text{VK}_h$. We now define some random variables.

Let ν be a random variable denoting the joint view of M and $V_1, \dots, V_{m_R}$ in a real execution of π. Similarly, $\nu^{(i)}$ will be the random variable denoting the output of hybrid simulator $\mathcal{H}_i$, $i = 1, 2, \dots$. For every "left" session $h \in [m_L]$, let $v_h^{(i)}$ denote the value committed to by M in phase-2 (i.e., extraction-preamble) of session h; and let $\mathbf{v}_h^{(i)}$ denote the value committed to by prover P_h in phase-3 (i.e., the DDN-commitment phase) of that session. Of course, $\mathbf{v}_h^{(i)} = 0$ for an honest prover. Define random variables $\tilde{v}_\ell^{(i)}, \tilde{\mathbf{v}}_\ell^{(i)}$ for right sessions $\ell \in [m_R]$, analogously. Thus, $\tilde{v}_\ell^{(i)}$ denotes the value committed to by V_ℓ in phase-2 of $\ell^{\rm th}$ right session; and $\tilde{\mathbf{v}}_\ell^{(i)}$ denotes the value committed to by M to V_ℓ in phase-3 of the same session on right, here $\ell \in [m_R]$. Finally, define $b_\ell^{(i)}$ to be a random boolean variable denoting whether in right-session ℓ, V_ℓ rejects ($b_\ell^{(i)} = 0$ and 1 otherwise) at the end of phase-3 (i.e., the DDN-commitment phase) in a simulation by $\mathcal{H}_i$.

Overall strategy of the proof. In our proof the key-idea is to ensure that $\forall \ell, \tilde{\mathbf{v}}_\ell^{(i)} \neq \tilde{v}_\ell^{(i)}$ while at the same time $\mathbf{v}_h^{(i)} = v_h^{(i)}$ ($\forall h$) with high probability.

We do this by designing a series of hybrid experiments $\mathcal{H}_i$ setting up $\mathbf{v}_h^{(i)} = v_h^{(i)}$ one-by-one for all left sessions h; it would be done while maintaining $\tilde{\mathbf{v}}_\ell^{(i)} \neq \tilde{v}_\ell^{(i)}$ for every right session in all the hybrid experiments with high probability. This would result in our final simulator using the Micciancio-Petrank method to succeed on left; whereas the adversary M will be forced to use the real witness

[13] In particular, this means that in second phase (extraction preamble phase), all commitments, challenges, and responses (i.e., openings) are valid; and during the fourth (i.e., opening) phase, $\mathcal{P}$ confirms that all openings are valid and that $v_0^{i,j} + v_1^{i,j} = v$ mod p for all values of i, j.

due to the aforementioned condition on right. We start by presenting our first hybrid.

Simulator $\mathcal{H}_0$. This simulator is provided with auxiliary inputs $y_\ell \in R_L(x_\ell)$ for all left statements x_ℓ for $\ell = 1, \ldots, m_L$. Let γ denote the uniformly chosen random tape of $\mathcal{H}_0$. The simulator starts interacting with $M(\boldsymbol{x}, z)$, where z is M's auxiliary input and $\boldsymbol{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_{m_L})$. On left, $\mathcal{H}_0$ acts as honest provers $P_1, \ldots, P_{m_L}$ (with independent and uniform random tapes) using inputs $y_1, \ldots, y_{m_L}$. On right, $\mathcal{H}_0$ acts as honest verifiers $V_1, \ldots, V_{m_R}$ (with independent and uniform random tapes). When M halts, $\mathcal{H}_0$ outputs the (joint) view of M and all $V_\ell, \ell \in [m_R]$, and halts. Recall that ν denotes the joint view in a real execution of π, and $\nu^{(0)}$ is the output of $\mathcal{H}_0$. The simulation is perfect, and so $\nu \equiv \nu^{(0)}$. Because we use a perfectly binding commitment scheme, values $\tilde{v}_\ell^{(0)}, \tilde{\mathbf{v}}_\ell^{(0)}$ are well defined. Let p_0 be the probability that there exists a right session ℓ such that $\left(\tilde{v}_\ell^{(0)} = \tilde{\mathbf{v}}_\ell^{(0)}\right)$ conditioned on the occurrence of the event "V_ℓ accepts".

Claim. $p_0 \leq \mu(\kappa)$

Proof. Contrary to the claim, suppose that $p_0 \geq 1/s(\kappa)$ for some polynomial $s(\cdot)$. Hence, for a non-negligible fraction of random tapes γ, it holds that for one of the right-sessions (say ℓ^{th}) M succeeds in setting $\tilde{v}_\ell^{(0)} = \tilde{\mathbf{v}}_\ell^{(0)}$, *and* V_ℓ accepts at the end of DDN-commitment phase of right-session ℓ (i.e., $b_\ell^{(0)} = 1$). We construct two machines M^*, M^*_{DDN}, and use them to break the semantic security of the commitment scheme denoted by the extraction preamble.

Machine M^* incorporates $M(\boldsymbol{x}, z)$ and interacts with it exactly as $\mathcal{H}_0$ except for the following two differences. First, in the ℓ^{th}-right-session, V_ℓ does not execute the extraction-preamble internally; instead it receives the commitment from an outside party A. That is, it chooses two values $v'_0, v'_1 \in \mathbb{Z}_p$ uniformly at random, and sends them to the outside sender A (of the extraction-preamble[14]). A then commits to v'_b, where $b \in_R \{0, 1\}$ which M^* forwards to $M(\boldsymbol{x}, z)$ as part of V_ℓ. Second, as soon as the preamble (of ℓ^{th}-right-session) concludes, M^* outputs its complete internal state, denoted ST_{M^*}, and halts.

Next, we use M^* to construct a DDN-sender M^*_{DDN} as follows. M^*_{DDN} starts with state ST_{M^*} and continues the rest of the execution internally exactly as $\mathcal{H}_0$, except for the following difference. In the DDN-commitment phase of ℓ^{th}-right-session, instead of internally emulating the actions of a DDN-receiver, V_ℓ (which is an internal part of M^*_{DDN}) interacts with an external DDN-receiver $\mathcal{R}_{\text{DDN}}$. M^*_{DDN} halts as soon as this phase finishes.

Finally, to break the semantic security of the extraction-preamble, our adversary (say $\mathcal{A}_{\text{COM}}$) proceeds as follows. Given $M(\boldsymbol{x}, z)$, $\mathcal{A}_{\text{COM}}$ first acts as M^* to receive a commitment from external A. Once, this interaction is over, we have the state ST_{M^*} and hence the adversary M^*_{DDN}. (By construction, the execution

[14] Recall that we can look at the extraction-preamble as an interactive commitment scheme.

of M^*_{DDN} is identical to that of $\mathcal{H}_0$ up to the point where DDN-phase completes). Now $\mathcal{A}_{\text{COM}}$ interacts with M^*_{DDN} while acting as $\mathcal{R}_{\text{DDN}}$, and if the interaction is accepting, it applies the DDN-extractor, E_{DDN}, to M^*_{DDN} and outputs whatever E_{DDN} outputs.

M^*_{DDN} is a machine that succeeds in committing to v'_b with probability $\geq p_0$ over the randomness of whole experiment. It follows that for at least $p_0/2$ fraction of views ST_{M^*}, M^*_{DDN} (using $M(\boldsymbol{x}, z)$) successfully commits the value v'_b to $\mathcal{R}_{\text{DDN}}$ with probability at least $p_0/2$. Thus, from the properties of the DDN-commitment scheme, we conclude that E_{DDN} extracts v'_b with probability $p_0/2 - \mu(\kappa)$ by running in expected polynomial time. Hence, $\mathcal{A}_{\text{COM}}$ can guess b with probability $p_0/2(p_0/2 - \mu(\kappa)) \geq p_0^2/8$ contradicting the semantic security of the extraction-preamble. □

Before proceeding further with the proof, imagine the following hybrid experiment $\mathcal{H}'_1$: it is the same as $\mathcal{H}_0$ except that it also performs the extraction of KP/PRS-secrets v_h on left by running both main as well as look-ahead threads just like the KP/PRS-simulator.[15]

Note that $\mathcal{H}'_1$ simulates honest verifiers $V_1, \ldots, V_{m_R}$ on right, and runs *real* provers $P_1, \ldots, P_{m_L}$ on left of $M(\boldsymbol{x}, z)$ in executing all the threads. If extraction of KP/PRS-secrets fails, (i.e., KP/PRS-simulator gets "stuck") than $\mathcal{H}'_1$ aborts.

Recall that the "threads" of a KP/PRS-simulator are classified into three types: a *main* thread, look-ahead threads that share a prefix with the main thread, and look-ahead threads that do not share any prefix with the main thread. Furthermore, all these threads can be *ordered* by their finishing time: thread 1 is the one that finishes first, thread 2 is the one that finishes second, and so on.

Threads contain several left and right sessions. In each left session belonging to a thread, if the execution of that session reaches the DDN-commitment phase, that session will contain the first message FM^*. Each thread can contain at most m_L such first messages, and there are at most $N \in O((\beta m_L)^2)$ FM^*s that ever appear in an execution of $\mathcal{H}'_1$. Further, in any given thread, these FM^*s can be ordered by their order of appearance, and since each thread can be ordered as explained above, we have an implicit ordering on these first messages which we denote by $\text{FM}^*_1, \ldots, \text{FM}^*_N$.

Observe that instead of executing all look-ahead threads at once, it is possible to only execute look-ahead threads of $\mathcal{H}'_1$ up to a specific point (e.g., up to the point where a specific first message FM^*_i appears) and then from thereon stop running any look-ahead threads and just complete the main-thread from where it was left.

We are now ready to explain our next $3N + 1$ hybrid simulators: $\mathcal{H}_{i:0}, \mathcal{H}_{i:1}$, and $\mathcal{H}_{i:2}$ for $i = 1, \ldots, N$. Define $\mathcal{H}_{0:2}$ to be the same as $\mathcal{H}_0$.

[15] Values $v_h, \mathbf{v}_h, \tilde{v}_\ell, \tilde{\mathbf{v}}_\ell$ are defined for session h of the *main* thread. For look-ahead threads, we'll introduce a new variable when needed.

Simulator $\mathcal{H}_{i:0}$. This experiment is the same as $\mathcal{H}_{i-1:2}$ except that it runs look-ahead threads up to the point where FM_i^* gets generated. After this point, the experiment continues the execution of main-thread directly *without running any look-ahead threads* at all. Note that up to this point, KP/PRS-secret v_j must have been extracted for all left sessions j for which the extraction preamble concludes successfully (in *any* thread), with high probability; and the experiment aborts if this is not the case.

Simulator $\mathcal{H}_{i:1}$. Consider an execution of our previous simulator $\mathcal{H}_{i:0}$. Since the execution reaches to the point FM_i^*, $\mathcal{H}_{i:0}$ must have extracted the value committed to in the extraction preamble of the session to which FM_i^* belongs. Denote this value by e_i.

Simulator $\mathcal{H}_{i:1}$ is the same as $\mathcal{H}_{i:0}$ except that when creating FM_i^*, it commits to e^i instead of committing to 0^κ using uniform randomness λ_i. It uses (e_i, λ_i) to complete the DDN-commitment phase of this session when needed.

Note that e_i is extracted "correctly" (i.e., equals the value opened by M later on in this session) with high probability. $\mathcal{H}_{i:1}$ aborts if this is not the case.

Simulator $\mathcal{H}_{i:2}$. This simulator is the same as $\mathcal{H}_{i:1}$ except that in all sessions j (across all threads) that belong to FM_i^*, if phase-5 is ever reached, it uses the Micciancio-Petrank simulator along with "trapdoor" (e_i, λ_i) defined in previous hybrid-simulator to succeed in this phase.

Our final simulator-extractor will use $\mathcal{H}_{N:2}$, to construct the final view. Due to space constraints, an analysis of above hybrid simulators is provided in the full version of this paper. We move on to present our final simulator.

The Final Simulator-extractor $\mathcal{S}$. For succinctness, let us denote $\mathcal{H}_{N:2}$ by $\mathcal{H}_2$. Our simulator-extractor $\mathcal{S}$ works as follows. It first runs the hybrid simulator $\mathcal{H}_2$ to produce a joint view $\nu^{(2)}$. The statements in the right executions (in $\nu^{(2)}$) are $\tilde{x}_1, \ldots, \tilde{x}_{m_R}$. For each right session $\ell \in [m_R]$, if V_ℓ accepts the proof and $\forall h \in [m_L]$ $\tilde{\text{VK}}_\ell \neq \text{VK}_h$, the simulator $\mathcal{S}$ extracts (a witness) $\tilde{y}_\ell \in R_L(\tilde{x}_L)$. The extraction for each such ℓ is performed *one by one*, as follows.

1. First, $\mathcal{S}$ defines an adversarial machine $\mathcal{A}_{\text{DDN}}^{(\ell)}$ as follows. $\mathcal{A}_{\text{DDN}}^{(\ell)}$ incorporates $M(\boldsymbol{x}, z)$ and proceeds exactly as $\mathcal{H}_2$ by internally simulating all the honest parties, except for the part of V_ℓ in the *main* thread which receives the phase-3 DDN-commitment. $\mathcal{A}_{\text{DDN}}^{(\ell)}$ terminates the execution after sending the knowledge-determining-message (KDM) to the external receiver. Now, $\mathcal{S}$ uses the (guaranteed) extractor which can work on this prefix (up to the KDM) to extract the value committed to by $\mathcal{A}_{\text{DDN}}^{(\ell)}$ in view $\nu^{(2)}$. Let the extracted value be u ($\mathcal{S}$ aborts if extraction fails).
2. If $u = \tilde{v}_\ell^{(2)}$, $\mathcal{S}$ aborts the extraction and halts. Otherwise, it defines a new machine $\mathcal{A}_{\text{MP}}^{(\ell)}$ as follows. $\mathcal{A}_{\text{MP}}^{(\ell)}$ is exactly as $\mathcal{H}_2$ that incorporates $M(\boldsymbol{x}, z)$ and all the simulated honest parties internally, except for the part of V_ℓ in the main thread which receives the phase-5 (i.e., Micciancio-Petrank) proof. $\mathcal{A}_{\text{MP}}^{(\ell)}$

is then a Micciancio-Petrank prover. It then applies the extractor guaranteed for such a prover, to extract a value $\tilde{y}_\ell$ – supposedly a witness for $\tilde{x}_\ell$ (repeat this procedure to obtain $\tilde{y} = \{\tilde{y}_1, \ldots, \tilde{y}_{m_R}\}$). It then outputs $\tilde{y}$ and halts.

A few remarks are in order. First, let us mention why we need to execute the first step involving $\mathcal{A}^{(\ell)}_{\text{DDN}}$, and why not directly execute the second step and extract $\tilde{y}_\ell$ using $\mathcal{A}^{(\ell)}_{\text{MP}}$. This is done in order to ensure that the extraction procedure has expected polynomial running time. Because otherwise, if $\tilde{\mathbf{v}}^{(2)}_\ell = \tilde{v}^{(2)}_\ell$ (even if with only negligible probability – equation **??**), the extraction procedure would never halt. As a result, the running time of $\mathcal{S}$ will not be bounded by any polynomial. Extracting u ($=\tilde{\mathbf{v}}^{(2)}_\ell$) using $\mathcal{A}^{(\ell)}_{\text{DDN}}$ allows $\mathcal{S}$ to abort whenever it is in this case.

Second, a subtlety in constructing $\mathcal{A}^{(\ell)}_{\text{DDN}}$ (and $\mathcal{A}^{(\ell)}_{\text{MP}}$ as well) is worth mentioning here. $\mathcal{A}^{(\ell)}_{\text{DDN}}$ acts as $\mathcal{H}_2$ internally and hence executes various "threads of execution" which may share a prefix with the main thread. When $\mathcal{A}^{(\ell)}_{\text{DDN}}$ interacts with the external receiver, it may define parts of some look-ahead threads. If the KDM did not appear in the shared prefix, $\mathcal{H}_2$ will have to *internally continue* the execution of these look-ahead threads who share a prefix with the main-thread (defined by the external receiver). The fact that the protocol is *public coin* up to the KDM, allows $\mathcal{H}_2$ to do that if required. Thus $\mathcal{A}^{(\ell)}_{\text{DDN}}$ (and for the same reason, $\mathcal{A}^{(\ell)}_{\text{MP}}$) is indeed well defined. From here, deriving our main theorem (Theorem 1) is not hard. Due to space constraints, this proof appears in the full version of this paper.

5 Efficiency

The Actual Cost. It is easy to see that the additional overhead incurred by the new prover and verifier, is dominated by three steps (overhead from all other steps is a *small* additive constant). First overhead is β^2 exponentiations (in $\mathbb{Z}_q$) due to the extraction-preamble.[16] The second overhead is due to the DDN-commitment phase, which as we discuss shortly, is $\kappa \cdot \tilde{O}(\log^2 \kappa)$ exponentiations. Finally, the last overhead is due to the Micciancio-Petrank transformation, which is r exponentiations, where r is the round complexity of π_{HV}. As $\beta \in \tilde{O}(\log \kappa)$, it follows that the additional overhead incurred by each party is (at most) $r + \kappa \cdot \tilde{O}(\log^2 \kappa)$ exponentiations in $\mathbb{Z}_q$.

The overhead in DDN-commitments is as follows. The cost is dominated by the following steps (overhead from all other steps is a *small* additive constant). First costly operation is the execution of SZKAOK, which requires $\omega(1)$ exponentiations. The next (in fact, the main) costly operation is the execution of step 3. This involves performing $\kappa \cdot (1 + \log \kappa) \cdot 2$ executions of $\text{BCK}^{\|}$. As $\text{BCK}^{\|}$ repeats BCK, in parallel, β times, and each BCK has an overhead of constant (less than 10) exponentiations in $\mathbb{Z}_q$, it follows that the overall overhead is $\kappa \cdot \tilde{O}(\log^2 \kappa)$ exponentiations in $\mathbb{Z}_q$.

[16] This overhead is only β exponentiations, if one chooses RK/KP-type preamble.

Cost for Tag-based Non-malleability. Historically, the verification key VK used in DDN-commitment protocol, is also called an *identity* or *tag*. Currently, the size of this tag is κ. If identities (or tags) are given to exist, then the first and the last steps of the protocol are unnecessary (and hence are not executed). Non-malleability in such cases requires the extraction of witness only when the adversary does not copy the tag entirely, and is called "tag-based" non-malleability. If tags of shorter length are possible, it results in more efficient protocols. The two mainly cited reasons for justifying this notion are the following ones. First, for some applications, it may be reasonable to assume that all parties have unique identities. As there are only polynomially many parties in real world protocols, they can all be represented by using tags of length at most $\omega(\log \kappa)$. Second, non-malleable protocols are typically used as building blocks in larger protocols. The execution of these larger protocols, may somehow, result in establishing tags for this building block.

For tag-based non-malleability, assuming the tag-length, $|\text{VK}|$, is $\tilde{O}(\log \kappa)$ – which we believe is reasonable – the overhead in the DDN-commitment phase would only be $\tilde{O}(\log^2 \kappa)$. And thus, the overhead incurred by each party in our transformation would be at most $r + \tilde{O}(\log^2 \kappa)$.

We would like to mention here that our transformed protocol is very suitable for the employment of preprocessing and batching techniques.

Efficient CNMZK *in the* BPK *Model.* In the full version of this paper we show that our tag-based non-malleable protocols lead to first truly efficient constructions in the BPK model [11]. This model has been used in sequence of papers [26–28] to initially achieve round and computationally efficient concurrent zero knowledge and later constant-round concurrent non-malleable zero-knowledge [29, 30].

We give an efficiency preserving compiler for obtaining CNMZK arguments from any HVSZK argument π' in the (true) BPK model. We obtain these results by applying our efficient tag-based constructions of CNMZK arguments in the plain model. (When coupled with a proper π', this gives efficient constructions of comparable efficiency.) By efficient we mean that the round complexity of the new protocol is $r + \tilde{O}(\log \kappa)$ while the additional computational overhead incurred by each party would be at most $r + \tilde{O}(\log^2 \kappa)$.

References

1. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. 17th STOC. (1985) 291–304
2. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: Proc. 44th FOCS. (2003) 394–403
3. Lindell, Y.: Lower bounds for concurrent self composition. In: Theory of Cryptography Conference (TCC). Volume 1. (2004) 203–222
4. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. FOCS 2006. Full version on Cryptology ePrint Archive report. (2006) `http://eprint.iacr.org/`.
5. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing **30**(2) (2000) 391–437 (electronic) Preliminary version in STOC 1991.

6. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening zero-knowledge protocols using signatures. In: EUROCRYPT. (2003) 177–194
7. MacKenzie, P., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Advances in Cryptology – Eurocrypt '04. 382–400
8. Gennaro, R.: Multi-trapdoor Commitments and Their Applications to Proof s of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks. In: Advances in Cryptology – Crypto '04. 220–236
9. Damgård, I., Nielsen, J.B., Orlandi, C.: On the necessary and sufficient assumptions for uc computation. In: TCC. LNCS (2010)
10. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO. (1991) 129–140
11. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge. In: Proc. 32th STOC. (2000) 235–244
12. Micciancio, D., Petrank, E.: Simulatable commitments and efficient concurrent zero-knowledge. In: EUROCRYPT. (2003) 140–159
13. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: TCC. (2008) 571–588
14. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: Proc. 37th STOC. (2005)
15. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: CRYPTO ' 2001. (2001) 566–598
16. Mohassel, P., Franklin, M.K.: Efficiency tradeoffs for malicious two-party computation. In: Public Key Cryptography. (2006) 458–473
17. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: EUROCRYPT. (2007) 79–96
18. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: EUROCRYPT. (2007) 52–78
19. Goyal, V., Mohassel, P., Smith, A.: Efficient two party and multi party computation against covert adversaries. In: EUROCRYPT. (2008)
20. Chase, M., Lysyanskaya, A.: Simulatable vrfs with applications to multi-theorem nizk. In: CRYPTO. (2007) 303–322
21. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: EUROCRYPT. (2006) 339–358
22. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press
23. Schnorr, C.P.: Efficient identification and signatures for smart cards (abstract). In: EUROCRYPT. (1989) 688–689
24. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Eurocrypt '99. (1999) 415–432
25. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS. (2002) 366–375
26. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In: Advances in Cryptology – Crypto '04
27. Di Crescenzo, G., Visconti, I.: Concurrent zero knowledge in the public-key model. In: Proceedings of ICALP 2005. 816–827
28. Visconti, I.: Efficient zero knowledge on the internet. In: ICALP 2006
29. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-round concurrent nmwi and its relation to nmzk. Technical Report ECCC Report TR06-095, ECCC (2006)
30. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In: ICALP 2008. (2008)