

Adaptive Security of Concurrent Non-Malleable Zero-Knowledge

Zhenfu Cao

Computer Science&Engineering Department
Shanghai Jiao Tong University, China
zfc@cs.sjtu.edu.cn

Zongyang Zhang*

Computer Science&Engineering Department
Shanghai Jiao Tong University, China
zongyangzhang@gmail.com

Yunlei Zhao†

Software School
Fudan University, China
ylzhao@fudan.edu.cn

Abstract

A zero-knowledge protocol allows a prover to convince a verifier of the correctness of a statement without disclosing any other information to the verifier. It is a basic tool and widely used in many other cryptographic applications. However, when stand-alone zero-knowledge protocols are used in complex environments, e.g., the Internet, the basic properties may not be sufficient. This is why researchers considered security of zero-knowledge protocols under concurrent composition and man-in-the-middle attacks. Moreover, it is very likely that an adversary might break computers that run the protocol and get internal information of the parties. It is thus necessary to take account of the security of zero-knowledge protocols when adaptive corruptions are allowed.

Previous adaptively secure zero-knowledge protocols work either in a stand-alone setting, or in a concurrent setting with trusted setup assumptions. In this paper, we study adaptive security of zero-knowledge protocols under both concurrent self composition and man-in-the-middle attacks in the plain model (i.e., without any set-up assumptions). We provide a construction of adaptively secure concurrent non-malleable zero-knowledge proof/argument for every language in NP.

Keywords: Zero-knowledge protocol, concurrent non-malleability, adaptive corruption, commitment schemes

1 Introduction

Zero knowledge proofs, introduced by Goldwasser, Micali and Rockoff [GMR85], allow a prover to convince a verifier the validity of a statement without disclosing any other information to him. It was shown that every NP language has a zero-knowledge proof system [GK96]. Zero-knowledge proofs are widely used in many cryptographic applications and are one of the most fundamental cryptographic building blocks. As application execution environments change from one to another, stand-alone zero-knowledge proofs might fail to satisfy security requirements in these various settings.

Consider a setting where there are many instances of protocols which are invoked at arbitrary times. Here, many verifiers are receiving proofs from various independent provers, and trying to collude together to learn something non-trivial from the provers. It was shown that a stand-alone zero-knowledge protocol fails to preserve the zero-knowledge property in the above setting [DNS98]. Thus, researchers considered the notion of concurrent zero-knowledge. Another setting is that there are man-in-the-middle attacks. A man-in-the-middle adversary may convince a verifier of a statement that it otherwise cannot do by interacting with an honest prover. The notion of non-malleable zero-knowledge proofs was first introduced by Dolev, Dwork and Naor [DDN00] to capture security requirements in this setting.

The notion of concurrent non-malleable zero-knowledge (CNMZK) considers both of the above settings. An adversary may interact with many provers while playing the role of the verifier, and simultaneously interact with many verifiers while playing the role of the prover. Barak, Prabhakaran

*Partial work is done while the author is visiting Singapore Management University.

†Partial work is done while the author is visiting Singapore Management University.

and Sahai [BPS06] gave the first CNMZK argument in the plain model. Ostrovsky, Pandey and Visconti [OPV10] improved this result and gave more efficient instantiations of CNMZK arguments for a special class of NP languages. Lin, Pass, Tseng and Venkatasubramanian (LPTV) [LPTV10] presented the first construction of CNMZK proofs.

Previous CNMZK protocols in the plain model only considered settings where an adversary is allowed to control/corrupt parties before the start of protocol executions, but, is not allowed to corrupt parties during protocol executions. However, in reality, based on the information that was already gathered, adversaries (e.g., hackers, viruses, insiders) may break into computers, possibly while they are executing secure protocols. Thus, it is very necessary to model adaptive security of zero-knowledge protocols since this models realistic security threats better and so provides a better security guarantee. We call a zero-knowledge protocol secure in this setting an adaptively secure CNMZK, or CNMZK against adaptive adversaries. Previous results on adaptive security of zero-knowledge protocols either work in the stand-alone setting [Bea96, LZ09], or under concurrent composition with trusted setup assumptions [CF01, DN02, CLOS02, LZ09]. This raises the following intriguing question:

Does there exist a concurrent non-malleable zero-knowledge proof/argument system for every NP against adaptive corruptions in the plain model?

To design an adaptively secure protocol is always believed to be a more challenging work than designing its statically secure counterpart. Therefore, researchers tend to resort to other assumptions (e.g., parties have tamper proof hardware tokens [Kat07], or parties can reliably erase information [Lin09]) to overcome obstacles and simplify the design. As we focus on the plain model, we intend not to use these stronger physical assumptions. We will explain the main obstacles in detail in the next section and show how to overcome them step by step.

1.1 Our Results

In this work, we give a positive answer to the above question and show the following result.

Theorem 1.1. *Assume the existence of one-way functions. Then there exists a $\text{poly}(n)$ -round adaptively secure concurrent non-malleable zero-knowledge proof for every NP. Furthermore, assuming the existence of collision-resistant hash-functions, the round complexity is $\tilde{O}(\log n)$.*

As an additional contribution, we give a construction of adaptively secure CNMZK argument.

Theorem 1.2. *Assume the existence of one-way functions. Then there exists a $\text{poly}(n)$ -round adaptively secure concurrent non-malleable zero-knowledge argument for every NP. Furthermore, assuming the existence of collision-resistant hash-functions, the round complexity is $\tilde{O}(\log n)$.*

Other related work. Recently, Yao, Yung and Zhao [YYZ10] presented the first construction of CNMZK protocol with full adaptive input selection in the bare public-key model. They allow the inputs of both left and right interactions to be adaptively chosen by an adversary; moreover, the input to an interaction can be decided adaptively *at any time* during this interaction. Lin and Pass [LP11a] provided the first construction of CNMZK protocol with adaptive inputs in the plain model. Furthermore, the input of an interaction is adaptively chosen by an adversary *at the outset* of each interaction. They called this notion adaptive CNMZK, which is a bit confusing with our notion of adaptively secure CNMZK. The main difference is that we focus on adaptive corruptions by an adversary, whereas Lin and Pass focused on adaptive inputs selected by an adversary. We believe that our technique might be extended to design adaptively secure CNMZK protocols with adaptive input selection based on the work of Lin and Pass [LP11a].

Techniques. Our CNMZK protocol is based on the LPTV protocol [LPTV10]. We first recall the structures of the LPTV protocol. This protocol roughly contains three phases. In the first phase, called the *preamble phase*, the verifier commits to a random value (called trapdoor) using a concurrently extractable commitment scheme CECOM. In the second phase, called the *commit phase*, the prover commits to a witness of the proof statement using both a CECOM and robust non-malleable commitment scheme NMCOM. Finally, in the third phase, called the *proof phase*, the prover proves using a stand-alone zero-knowledge protocol that it has either committed to a valid witness or a valid trapdoor in the

commit phase. To prove security, the LPTV simulator uses rewindings to extract out trapdoors in the preamble phases and then commits to the trapdoors in the commit phases. Using the “fake witnesses” (i.e., decommitment information to the commitments in the commit phases), the simulator is able to run the zero-knowledge proof in a straight-line manner in the proof phases. For all right accepting interactions, the LPTV simulator again uses rewindings to extract out the witnesses committed to by the adversary (from CECom in the commit phases).

When considering adaptive corruptions to verifiers on the right, the LPTV simulator can be directly adjusted to handle this case. As the LPTV simulator follows the honest verifier strategy in all right interactions and the verifier has no secret information, the simulator just provides randomness for the simulated verifier when a verifier is corrupted. However, when considering adaptive corruptions to provers on the left, several problems arise. First, if a prover is corrupted after completion of the CECom in the commit phase, the simulator is given a witness w to the proof statement x , and now it has to provide the adversary with the randomness of the simulated prover in the CECom . Recall that the LPTV simulator commits to a trapdoor instead of the witness w , and the commitment CECom is binding. Now the LPTV simulator gets stuck in explaining the commitment as to w . Second, the same problem arises for NMCom in the commit phase. The LPTV simulator again gets stuck if the adversary corrupts a prover after the completion of NMCom , since it committed to a trapdoor and has to explain the commitment as to w . Finally, upon corruption of a prover of the zero-knowledge proof in the proof phase, the LPTV simulator has to explain a proof as one generated using the real witness, whereas it is actually generated using a “fake witness”. The LPTV simulator again cannot handle this case.

Our idea for circumventing the above three problems can be described as follows:

- For the first problem, we rely on a concurrently extractable commitment CECTCom which is also concurrent trapdoor. The trapdoor property guarantees that there exists a simulator for the commitment, such that knowing the trapdoor, it is able to open the commitment to arbitrary values. Using CECTCom , the CNMZK simulator is able to explain a commitment to a trapdoor (i.e., the value committed by an adversary in the preamble phase) as to the real witness w . Our construction of CECTCom uses a concurrent trapdoor commitment scheme CTCom which is statistically binding. At first sight, it seems that it is impossible to design CTCom against fully adaptive corruptions in the plain model, since we have to achieve seemingly contradicting goals that, on one hand, even an infinitely powerful committer is not able to equivocate the commitment, and on the other hand, an expected PPT simulator can equivocate the commitment. However, we overcome this obstacle with the help of the technique of the work [CO99]. Roughly, we rely on the property of Naor’s commitment scheme which is equivocal if the result of the first message can be programmed/controlled by the simulator. We let this message be generated by a coin-tossing protocol between the committer and the receiver. Through this way, the simulator is able to hand the corruption of a committer after execution of the CTCom (and then CECTCom).
- For the second problem, we rely on a robust non-malleable commitment scheme NMCTCom that is concurrent trapdoor. The NMCTCom is based on the non-malleable commitment in [LPV08] and a concurrent trapdoor commitment scheme CTCom .¹ Using similar analysis as above, the trapdoor property of NMCTCom can be used to handle adaptive corruptions. Another problem related with NMCTCom is how to handle adaptive corruptions during special-sound witness-indistinguishable (\mathcal{WZ}) proofs.² The simulator has to interpret the proof generated using a “fake witness” as one generated using a real witness. Here we resolve the above problem by requiring an honest prover to commit using CTCom in each of the special-sound proofs.³ Through this way, we ensure that the simulator is able to provide all the corresponding randomness upon corruption of a committer after the execution of NMCTCom .
- For the third problem, we rely on an adaptive instance-dependent scheme AIDCom proposed by

¹A possible efficient way to design an adaptively secure NMCTCom is to modify the recent constant-round concurrent non-malleable commitment schemes [LP11b, Goy11] and make them adaptively secure. However, we cannot work it out and left it as an open problem.

²In the non-malleable commitment scheme [LPV08], the receiver first computes a random image s of a one-way function, then the committer commits to its message and proves using a sequence of special-sound \mathcal{WZ} proofs that it knows either the opening of the commitment or the preimage of s .

³The proof system is identical to Blum’s basic protocol for Hamiltonicity. It consists of three (or four) rounds. The prover generates a commitment in the first round, the verifier then sends a random challenge and the prover responds according to the challenge.

Lindell and Zorosim [LZ09]. An instance-dependent commitment scheme is a commitment whose properties depend on whether the instance in question is in the language or not. Lindell and Zorosim uses AIDCom to construct (stand-alone) zero-knowledge *proof* systems against adaptive corruptions. We also use AIDCom to handle adaptive corruptions for the zero-knowledge *proof* in the proof phase. However, to make our simulation go through, we are unable to apply the analysis in [LZ09]. We have to explain a proof generated using a “fake witness” as one generated using a real witness.

The approaches to solving the above three problems result in a new problem. In the simulation, we have to run the extractor of CECOM in the preamble phases, in addition to the concurrent trapdoor simulator of both CECTCom and NMCTCom in the commit phases. It seems that we have to compose the (possibly conflicting) individual rewinding strategies and present a complicated analysis. In order to get rid of this obstacle, we combine part of commitment CECTCom and NMCTCom (i.e., the part for extraction) with CECOM in the preamble phase, and need only a new uniform rewinding strategy. In the following, we will present constructions of protocols CECTCom and NMCTCom of which the properties satisfy this simulation strategy.

Organization. We present the definition of CNMZK in Section 2. In Section 3 we introduce all basic tools that we use in the construction of CNMZK. In Section 4, we give a construction of CNMZK proof systems for all NP.

2 Preliminaries and Definitions

Let \mathbb{N} be the set of all positive integers. For any integer $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, 2, \dots, n\}$. Let $\{0, 1\}^n$ be the set of n -bit strings. We assume familiarity with computational/statistical indistinguishability, interactive proofs, zero-knowledge, commitment schemes, and (strong) witness-indistinguishability.

2.1 Adaptively Secure Concurrent Non-Malleable Zero-Knowledge

Adaptively concurrent man-in-the-middle attack. Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive proof for NP language L with witness relation R_L . Consider a man-in-the-middle adversary \mathcal{A} that participates in many left and right interactions. Without loss of generality, suppose that at most $m = m(n)$ proofs take place. Prior to all interactions, all parties in the system receives as common input the security parameter in unary 1^n , and \mathcal{A} receives as auxiliary input $z \in \{0, 1\}^*$. The concurrent man-in-the-middle setting proceeds as follows. First, the input statements to honest provers, i.e., statements $x_1, \dots, x_m \in L \cap \{0, 1\}^n$, and the corresponding tags $\text{id}_1, \dots, \text{id}_m \in \{0, 1\}^{t(n)}$ are chosen. \mathcal{A} interacts with the honest prover \mathcal{P}_i with common input x_i and id_i while playing the role of a verifier. \mathcal{P}_i receives as local input the witnesses w_i such that $w_i \in R_L(x_i)$. These interactions are called “left” interactions. During the left interactions, \mathcal{A} can corrupt arbitrary honest provers. At any point, \mathcal{A} may adaptively choose a new statement \tilde{x}_i and tag $\tilde{\text{id}}_i$ and start a new proof with a verifier \mathcal{V}_i while acting as the role of a prover. These interactions are called the “right” interactions. Furthermore, during the right interactions, \mathcal{A} is able to corrupt arbitrary honest verifiers. Once a party is corrupted, its common input, random input, and the entire history of the messages sent and received by the party are already known to \mathcal{A} . We denote by \vec{X} the input vector (x_1, \dots, x_m) and $\vec{\text{ID}}$ the tag vector $(\text{id}_1, \dots, \text{id}_m)$. Let $\text{view}_{\mathcal{A}}(1^n, \vec{X}, \vec{\text{ID}}, z)$ be the view of the adversary \mathcal{A} in the above experiment, i.e., it consists of \mathcal{A} ’s random coins, all common inputs and the transcripts of all left and right interactions between \mathcal{A} and honest provers and verifiers, and all collected information of corrupted parties. Given a function $t = t(n)$, we use the notation $\{\cdot\}_{n, \vec{X}, \vec{\text{ID}}, z}$ as shorthand for $\{\cdot\}_{n \in \mathbb{N}, x_1, \dots, x_m \in L \cap \{0, 1\}^n, \text{id}_1, \dots, \text{id}_m \in \{0, 1\}^t, z \in \{0, 1\}^*}$.

Roughly, an interactive proof is adaptively secure CNMZK if for every man-in-the-middle adaptive adversary \mathcal{A} , there exists a PPT simulator-extractor that can simulate both the left and right interactions for \mathcal{A} , while outputting a witness for every statement proved by \mathcal{A} in the right interactions. We emphasize here that when a prover is corrupted (not at the outset of the protocol), the simulator is then entitled to the prover’s input and witness, and it need not compute the view of the adversary from scratch. It only need fill the heretofore unknown portions in the adversary’s view.

Definition 2.1 (Adaptively Secure Concurrent Non-Malleable Zero-Knowledge). An interactive proof $\langle \mathcal{P}, \mathcal{V} \rangle$ for membership in an NP language L with witness relation R_L is called *adaptively secure concurrent non-malleable zero-knowledge* with tags of length $t = t(n)$ if for every PPT man-in-the-middle adaptive adversary \mathcal{A} that participates in at most $m = m(n)$ concurrent executions, there exists a PPT simulator-extractor \mathcal{S} such that,

- The two ensembles $\{\mathcal{S}_1(1^n, \vec{X}, \vec{\text{ID}}, z)\}_{n, \vec{X}, \vec{\text{ID}}, z}$ and $\{\text{view}_{\mathcal{A}}(1^n, \vec{X}, \vec{\text{ID}}, z)\}_{n, \vec{X}, \vec{\text{ID}}, z}$ are computationally indistinguishable over the security parameter $n \in \mathbb{N}$, where $\mathcal{S}_1(1^n, \vec{X}, \vec{\text{ID}}, z)$ denote the first output of $\mathcal{S}(1^n, \vec{X}, \vec{\text{ID}}, z)$.
- Denote by $(\nu, (\tilde{w}_1, \dots, \tilde{w}_m))$ the output of $\mathcal{S}(1^n, \vec{X}, \vec{\text{ID}}, z)$. Let $(\tilde{x}_1, \dots, \tilde{x}_m)$ be the statements of right interactions in the view ν . Let $\tilde{\text{id}}_1, \dots, \tilde{\text{id}}_m$ be the identities of the right interactions in the view ν . For every $i \in [m]$, if the verifier \mathcal{V}_i is not corrupted, the transcript of the i th right interaction is accepting and $\tilde{\text{id}}_i \neq \text{id}_j$ for all $j \in [m]$, then \tilde{w}_i is the witness to membership \tilde{x}_i in the language $L \cap \{0, 1\}^n$ except with negligible probability, i.e., $R_L(\tilde{x}_i, \tilde{w}_i) = 1$. Otherwise, \tilde{w}_i is set to \perp .

3 Basic Tools

3.1 Concurrently Extractable Commitment Schemes

The notion of concurrently extractable commitment scheme is first introduced by Micciancio et al. [MOSV06]. This is an abstraction of the preamble stage of the concurrent zero-knowledge protocol of [PRS02]. Roughly, a commitment scheme is concurrently extractable if there exists an efficient extractor that is able to generate a view that is statistically indistinguishable with the view of a malicious committer in the commit phases, and moreover, extract the committed values from any valid commitments sent by the committer. In this paper, we will use a concurrently extractable statistically hiding commitment scheme CECom_{sh} and a concurrently extractable statistically binding commitment scheme CECom_{sb} .

3.2 Concurrent Trapdoor Commitment Schemes

Roughly, a trapdoor commitment is a commitment scheme with an additional property such that there exists a simulator, with knowledge of some trapdoor information, can overcome the binding property and open a commitment arbitrarily. We extend this notion to concurrent execution settings and define concurrent trapdoor commitment schemes.

On a high level view, our concurrent trapdoor commitment scheme is build upon the trapdoor commitment scheme in [CO99] and a concurrently extractable commitment scheme. The committer and the receiver together first run a coin-tossing protocol to generate a random string r . Then the committer commits to its value using Naor's commitment scheme with r as the first message. Let Com_{sb} be Naor's two-round statistically binding commitment scheme. Let CECom_{sh} be a concurrently extractable statistically hiding commitment scheme. The commitment scheme CTCom_{sb} is shown in Figure 1. Due to space constraints, the proof of CTCom_{sb} is shown in Appendix B.1.

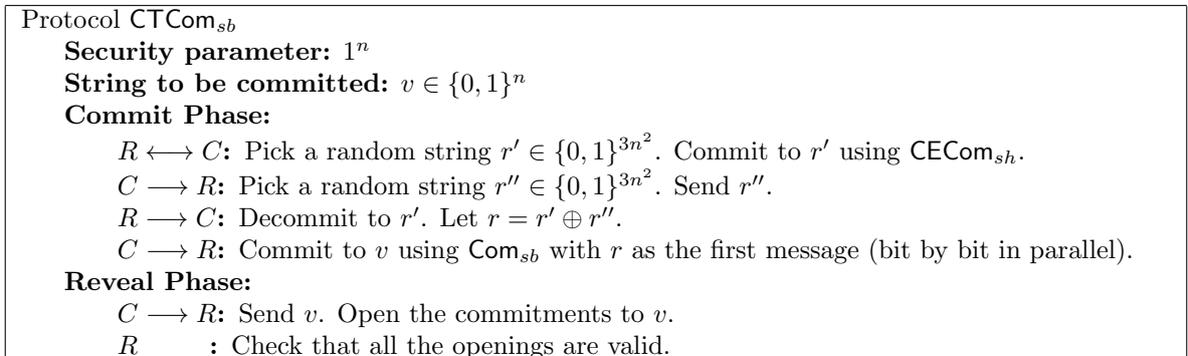


Figure 1: Concurrent trapdoor commitment scheme

To commit to an n -bit string v under scheme CTCom_{sb} , the receiver first commits to a random $3n^2$ -bit string r' using CECom_{sh} . Then the committer sends a random string r'' . Next the receiver opens its random string r' and the committer commits to v using Com_{sb} with $r' \oplus r''$ as the first message. In the reveal phase, the committer just opens its commitment to v .

It would be worth noting explicitly that in the reveal phase, all the randomness used in the commit phase will be revealed. This property is very useful when proving adaptive property of the concurrent non-malleable zero-knowledge protocol in Section 4.

3.3 Concurrently Extractable and Concurrent Trapdoor Commitment Schemes

On a high level view, our concurrently extractable and concurrent trapdoor statistically binding commitment scheme CECTCom_{sb} follows the structure of the concurrently extractable commitment scheme in [PRS02, MOSV06]. We simply replace the commitment used by the committer with a concurrent trapdoor one. The commitment scheme is shown in Figure 2. Due to space constraints, the proof is shown in Appendix B.2.

<p>Protocol CECTCom_{sb}</p> <p>Security parameter: 1^n</p> <p>Inputs: a value $v \in \{0, 1\}^n$</p> <p>Commit Phase:</p> <p>$C \rightarrow R$: Generate $n\ell$ pairs of random n-bit strings $(\alpha_{i,j}^0, \alpha_{i,j}^1)$, $i \in [n], j \in [\ell]$ such that for all i, j $v = \alpha_{i,j}^0 \oplus \alpha_{i,j}^1$. Commit to v and all $n\ell$ pairs strings using CTCom_{sb} one by one.</p> <p>For $j = 1$ to ℓ:</p> <p>$R \rightarrow C$: Send a random n-bit string $e_j = (e_{1,j}, \dots, e_{n,j})$.</p> <p>$C \rightarrow R$: Open the corresponding $\alpha_{i,j}^{e_{i,j}}$ for all $i \in [n]$.</p> <p>Reveal Phase:</p> <p>The committer opens all the remaining $n\ell + 1$ commitments. The receiver checks the correctness of openings and $v = \alpha_{i,j}^0 \oplus \alpha_{i,j}^1$ for all $i \in [n], j \in [\ell]$</p>
--

Figure 2: Concurrently extractable and concurrent trapdoor commitment scheme

Let $\ell = \ell(n)$ be any super-logarithmic function. To commit to an n -bit string v under scheme of CECTCom_{sb} , the committer generates $n\ell$ pairs of random n -bit strings such that each pair is a $(2, 2)$ share of the committed value v , i.e., $v = \alpha_{i,j}^0 \oplus \alpha_{i,j}^1$ for all $i \in [n], j \in [\ell]$. The committer then commits to v and each of the $2n\ell$ strings in parallel using CTCom_{sb} . This is followed by ℓ rounds of interactions. In the j th interaction, the receiver sends a random n -bit challenge $e_j = (e_{1,j} \dots e_{n,j})$ and the committer opens the commitment of $\alpha_{i,j}^{e_{i,j}}$ for all $i \in [n]$. In the reveal phase, the committer opens all the remaining $n\ell + 1$ commitments, and the receiver checks that all the openings are correct and the opened values satisfy $v = \alpha_{i,j}^0 \oplus \alpha_{i,j}^1$ for all $i \in [n], j \in [\ell]$.

We also point out a nice property of CECTCom_{sb} that in the reveal phase, all the randomness used in the commit phase will be revealed. This property is very useful when proving adaptive property of the concurrent non-malleable zero-knowledge protocol in Section 4.

3.4 Non-Malleable Concurrent Trapdoor Commitment Schemes

Roughly speaking, a commitment is non-malleable if an adversary cannot transform a commitment to a value into a commitment to a related value. This definition is introduced by Dolev, Dwork and Naor [DDN00]. A robust non-malleable commitment is non-malleable with respect to any protocol that has a small round complexity (i.e., less than $\tilde{O}(\log n)$ rounds). This definition is introduced by Lin and Pass [LP09]. A non-malleable concurrent trapdoor commitment is a commitment that is non-malleable and concurrent trapdoor.

Special-sound proofs. A k -round interactive proof for the language $L \in \text{NP}$ with witness relation R_L is *special-sound with respect to* R_L if the following holds: there exists a deterministic polynomial-time procedure that can extract a witness with overwhelming probability given a randomly sampled $(k - 2)$ -message prefix $\vec{\alpha}$ of the protocol and two independent accepting completions of the prefix $(\vec{\alpha}, \beta, \gamma)$ and

$(\vec{\alpha}, \beta', \gamma')$. By parallel running Blum's basic protocol for Hamiltonicity, we get a 4-round special-sound \mathcal{WI} proofs for NP assuming the existence of one-way functions. Moreover, if the commitment scheme used by the protocol [Blu86] is a CTCom_{sb} , then it becomes a k -round special-sound \mathcal{WI} proofs, where k depends on the round complexity of CTCom_{sb} . In the following, unless explicitly mentioned, when we say special-sound \mathcal{WI} proofs, we actually mean the above k -round one.

We construct a concurrent non-malleable concurrent trapdoor commitment scheme CNMCTCom in Figure 3. The protocol is based on the commitment in [LPV08] and adapted to our setting. The scheduling in Stage 3 of the protocol is shown in Figure 4. Due to space constraints, the proof is shown in Appendix B.3.

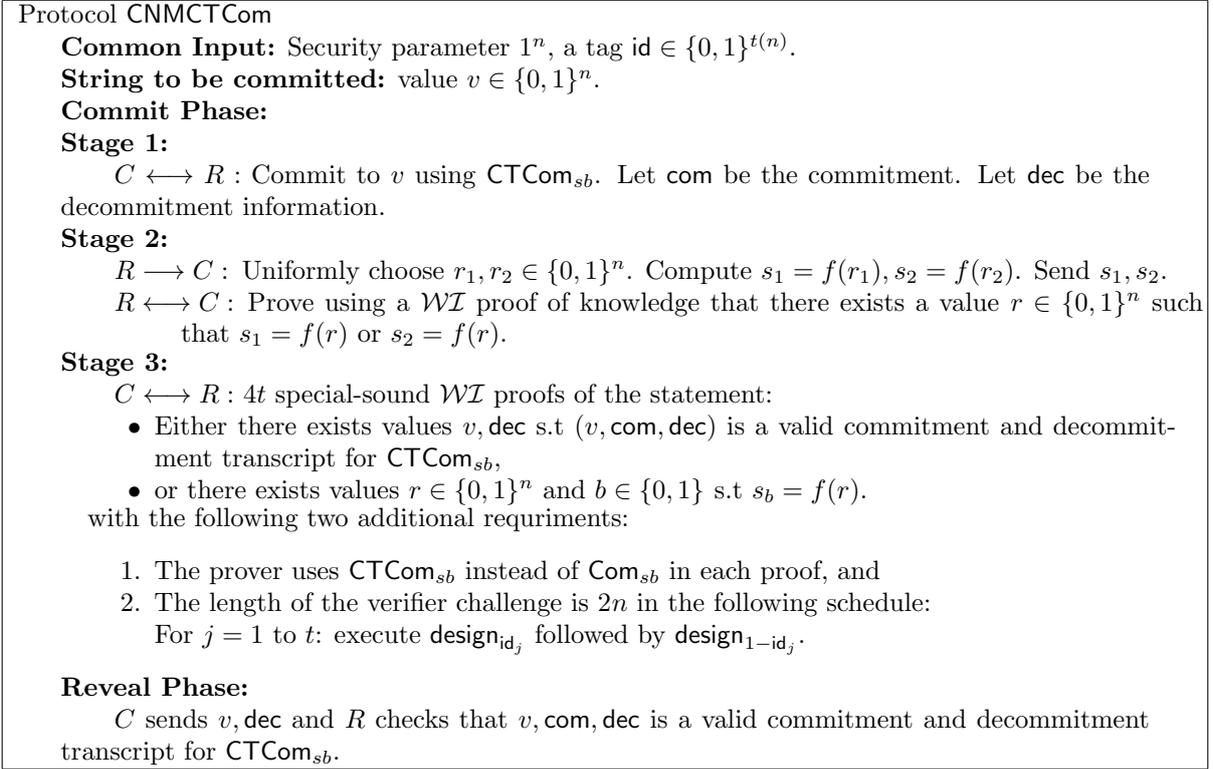


Figure 3: Concurrent non-malleable concurrent trapdoor commitment scheme

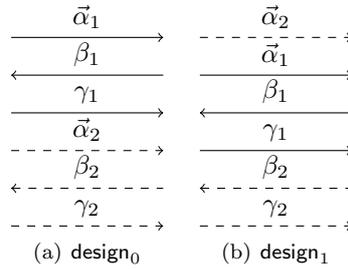


Figure 4: Two schedules

Extension to $\mathcal{O}(\log n)$ -round non-malleable concurrent trapdoor commitment schemes. Just as the technique in [LPV08], we show how to construct a $\mathcal{O}(\log n)$ -round commitment scheme that is stand-alone non-malleable and concurrent trapdoor using any $\mathcal{O}(n)$ -round commitment scheme that is concurrent non-malleable concurrent trapdoor. The protocol NMCTCom is constructed as follows. To commit to a value $v \in \{0, 1\}^n$, a committer chooses random strings $v_1, \dots, v_n \in \{0, 1\}^n$ such that

$v = v_1 \oplus \dots \oplus v_n$. If $\text{id} \in \{0, 1\}^n$ is the tag of the protocol NMCTCom , then the committer commits to v_i in parallel using CNMCTCom under tag (i, id_i) , where id_i is the i th bit of id . The protocol NMCTCom is a non-malleable and concurrent trapdoor commitment scheme. The non-malleability proof is much similar with that in [LPV08]. The trapdoor property follows from the concurrent trapdoor property of CNMCTCom . We defer the proofs to the full version.

Extension to non-malleable concurrent trapdoor commitment scheme robust w.r.t k -round protocols. Actually, the above $\mathcal{O}(\log n)$ -round non-malleable commitment scheme is robust w.r.t $\log n$ -round protocols, since it has at least $\log n$ rewinding slots. By choosing the parameter carefully, we can prove the following lemma.

Lemma 3.1. *Let $\ell(n)$ be a super-logarithmic function. Then there exists a $\mathcal{O}(\ell(n))$ -round concurrent trapdoor statistically binding commitment scheme that is robust w.r.t $\ell(n)$ -round protocols.*

3.5 Adaptive Instance-Dependent Commitment Schemes

Instance-dependent commitments were first introduced in [IOS97]. Roughly speaking, an instance-dependent commitment scheme is a commitment whose properties depend on whether the instance in question is in the language or not. Typically, it is defined for a language L as follows. Let x be a statement. If $x \in L$ then the commitment associated with x is computationally hiding, and if $x \notin L$ then the commitment associated with x is statistically binding. The adaptive instance-dependent commitment scheme $\text{AIDCom} = (\text{Com}, \text{Com}', \text{Adapt})$ was first introduced by Lindell and Zarusim [LZ09]. Roughly, it has the additional property that commitments are equivocal. It has a “fake” committing algorithm Com' which generates fake commitments. If knowing the witness to the statement, the fake commitment can be opened to arbitrary value by the adaptive opening algorithm Adapt ; otherwise, it cannot necessarily be opened to any value. Due to space constraints, the definition of adaptive instance-dependent commitment is shown in Appendix A.6.

4 Adaptively Secure Concurrent Non-Malleable Zero-Knowledge

In this section, we construct an adaptively secure concurrent non-malleable zero-knowledge proof for every NP language. Let CECom_{sh} be a concurrently extractable statistically hiding commitment scheme. Let CECTCom_{sb} be a concurrently extractable and concurrent trapdoor statistically binding commitment scheme. Let NMCTCom be a non-malleable concurrent trapdoor commitment scheme. Let $\text{AIDCom} = (\text{Com}, \text{Com}', \text{Adapt})$ be an adaptive instance-dependent commitment scheme for the language of Hamiltonicity. Let MBZKProof be a modification of Blum’s $\omega(1)$ -round zero-knowledge proof system for the language of Hamiltonicity, in which the prover commits to the adjacency matrices using AIDCom . The instance used by AIDCom is the statement proved in protocol MBZKProof .

Our concurrent non-malleable zero-knowledge protocol CNMZKProof is a variant of the LPTV protocol in [LPTV10]. The protocol CNMZKProof for an NP language L proceeds in six stages, given a security parameter n , a common input statement $x \in \{0, 1\}^n \cap L$, a tag $\text{id} \in \{0, 1\}^{t(n)}$, and a private input $w \in R_L(x)$ to the prover.

Stage 1: The verifier \mathcal{V} chooses a random string $r \in \{0, 1\}^n$ and commits to r using CECom_{sh} .

Stage 2: The prover \mathcal{P} commits to the witness w using CECTCom_{sb} .

Stage 3: The prover \mathcal{P} commits to the witness w using NMCTCom under tag id .

Stage 4: The prover \mathcal{P} commits to the witness w using NMCTCom under tag id again.

Stage 5: The verifier \mathcal{V} decommits its commitment in Stage 1 to value r .

Stage 6: The prover \mathcal{P} proves using MBZKProof that the the commitments in Stage 2, 3 and 4 all commits to the same value \tilde{w} , and either $\tilde{w} \in R_L(x)$ or $\tilde{w} = r$.

The main difference between our protocol CNMZKProof and the LPTV protocol is that we replace primitives used by the prover with corresponding “adaptive” ones on which the simulator relies to handle adaptive corruptions of provers. A formal description of the protocol CNMZKProof is shown in Figure 5.

Remark 1. For ease of exposition, we divided the protocol CNMZKProof into six stages. Actually, in the following proof (of simulation-extractability), we have to optimize the above protocol a bit to

Protocol CNMZKProof**Common Input:** an instance of $x \in L \cap \{0, 1\}^n$, a tag $\text{id} \in \{0, 1\}^{t(n)}$.**Auxiliary input to prover:** a witness w s.t. $(x, w) \in R_L$.**Stage 1:** \mathcal{V} uniformly chooses $r \in \{0, 1\}^n$. It commits to r using CECom_{sh} . Let \mathcal{T}_1 be the commitment transcript.**Stage 2:** \mathcal{P} commits to w using CECTCom_{sb} . Let \mathcal{T}_2 be the commitment transcript.**Stage 3:** \mathcal{P} commits to w using NMCTCom and tag id . Let \mathcal{T}_3 be the commitment transcript.**Stage 4:** \mathcal{P} commits to w using NMCTCom and tag id . Let \mathcal{T}_4 be the commitment transcript.**Stage 5:** \mathcal{V} decommits \mathcal{T}_1 to value r . \mathcal{P} aborts if the decommitment fails.**Stage 6:** $\mathcal{P} \leftrightarrow \mathcal{V}$: Denote by L' the language $\{(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)\}$, where an instance in L' satisfies: there exists \tilde{w} such that

- \tilde{w} is a valid opening of \mathcal{T}_2 ,
- and \tilde{w} is valid opening of \mathcal{T}_3 and \mathcal{T}_4 under tag id ,
- and $\tilde{w} \in R_L(x)$ or $\tilde{w} = r$.

Prove using MBZKProof that $(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r) \in L'$, i.e., both the prover and the verifier run a Cook-Levin reduction from L' to Hamiltonicity, and then invoke MBZKProof .

Figure 5: Adaptively secure concurrent non-malleable zero-knowledge proof for NP

make a better use of the simulation strategy of the LPTV protocol. Roughly, the reason is follows: in the simulation, we need run the extractor of CECom_{sh} in Stage 1, in addition to the concurrent trapdoor simulator of both CECTCom_{sb} and NMCTCom . It seems that we have to compose the (possibly conflicting) individual rewinding strategies and present a complicated analysis. In order to get rid of this obstacle, we combine the CECom_{sh} part in CECTCom_{sb} and NMCTCom with Stage 1, and need only a new uniform rewinding strategy. Recall that in the protocol CECTCom_{sb} , a committer first commits to its value and many independent shares of the value using CTCom_{sb} . In the the protocol NMCTCom , a committer first commits to its value using CTCom_{sb} , and also commits in the special-sound \mathcal{WZ} proof using CTCom_{sb} . The commitment CTCom_{sb} requires the receiver first commits to a random challenge using a CECom_{sh} . The CECom_{sh} part of CECTCom_{sb} in Stage 2 and of NMCTCom in Stage 3 and 4 can be merged into the Stage 1 of protocol CNMZKProof . So, we can view the Stage 1 consisting of four parts of parallel executions of CECom_{sh} .⁴

Remark 2. For ease of description, we actually use CNMCTCom instead of NMCTCom in the proof. In the proof, we invoke the concurrent trapdoor simulator of a variant of CNMCTCom , which is denoted by $\text{CNMCTCom}'$. $\text{CNMCTCom}'$ is the same as CNMCTCom except that all CECom_{sh} parts are executed in parallel beforehand. The reader is referred to Section B.3.1 for details. Here without confusion of notation, we instead use NMCTCom in the protocol.

Properties of MBZKProof . Note that we use a modification of Blum's $\omega(1)$ -round zero-knowledge proof MBZKProof in Stage 6 (For reference, Blum's basic protocol for Hamiltonicity is shown in Appendix D.). We first show it is still a zero-knowledge proof for the language of Hamiltonicity. The completeness is straightforward. Recall that in MBZKProof , the prover commits to the adjacency matrix of a random permutation of the graph using an adaptive instance-dependent commitment AIDCom instead of a statistically binding one. The soundness proof in [Blu86] relies on the binding property of the commitment, and AIDCom is statistically binding when an input instance is not a Hamiltonian graph, thus, the soundness property is preserved using almost the same analysis as in [Blu86]. On the other hand, the zero-knowledge simulation in [Blu86] relies on the hiding property of the commitment. Moreover, the AIDCom is computationally hiding when an input instance is a Hamiltonian graph. Thus, the zero-knowledge simulation is the same as that in [Blu86]. Lindell and Zarusim [LZ09] also designed a ZK simulator when adaptive corruptions are allowed. However, this ZK simulator does not suffice in the proof of the protocol CNMZKProof . We will design a different ZK simulator for MBZKProof which is especially suited to handle adaptive corruptions in our case.

Claim 4.1. *The protocol CNMZKProof is an adaptively secure concurrent non-malleable zero-knowledge proof system for NP.*

⁴Note that the third and the fourth part each consists of many parallel executions of CECom_{sh} .

Proof. We need to prove that the protocol CNMZKProof satisfies the following three properties: completeness, unconditional soundness, and simulation-extractability. Due to space constraints, we only give the construction of simulator-extractor \mathcal{S} as required by Definition 2.1, and the formal proof is shown in Appendix C.

Simulation-extractability. The definition of CNMZK requires a simulator-extractor \mathcal{S} that is able to simulate the view of a man-in-the-middle adaptive adversary \mathcal{A} , while simultaneously extracting the witnesses of statements proved in the right interactions. Roughly, \mathcal{S} proceeds as follows.

Simulation of right interactions: \mathcal{S} simply runs as honest verifiers in all right interactions.

Simulation of left interactions: In each protocol execution, \mathcal{S} proceeds as follows:

1. In Stage 1, \mathcal{S} first extracts a “fake witness” r , a challenge e_2 for CECTCom_{sb} , a challenge vector \vec{e}_3 for NMCTCom in Stage 3 and a challenge vector \vec{e}_4 for NMCTCom in Stage 4 from CECom_{sh} committed to by \mathcal{A} . (See Remark 1.)
2. In Stage 2, \mathcal{S} commits using the concurrent trapdoor simulator Ts^2 of CECTCom_{sb} . Denote by τ^2, aux^2 the simulated view and the auxiliary information, respectively.
3. In Stage 3, \mathcal{S} commits using the concurrent trapdoor simulator Ts^3 of NMCTCom .⁵ Note that knowing \vec{e}_3 in advance, the construction of the Ts^3 is straightforward. In more detail, \mathcal{S} simulates Stage 1 of NMCTCom by invoking the concurrent trapdoor simulator of CTCom_{sb} . \mathcal{S} simulates Stage 2 of NMCTCom by following the honest committer strategy. \mathcal{S} simulates Stage 3 of NMCTCom as follows. It uses the concurrent trapdoor simulator of CTCom_{sb} to generate the commitments and answer queries from the adversary. For simplicity and without loss of generality, we only present the simulation for Blum’s basic protocol for Hamiltonicity (which is used as the basic special-sound \mathcal{WI} proof). Denote by G the input.
 - (a) \mathcal{S} first picks a random permutation π on the vertices and commits to the adjacency matrix of random graph $G' = \pi(G)$ using the concurrent trapdoor simulator of CTCom_{sb} .
 - (b) The adversary picks a random bit challenge b .
 - (c) If $b = 0$, \mathcal{S} sends π along with the revealing of all commitments. That is, \mathcal{S} invokes the concurrent trapdoor simulator of CTCom_{sb} to open the commitments to corresponding value, i.e., for each entry $(i, j) \in G$, it opens the commitment corresponding to $(\pi(i), \pi(j))$ to 1, and for all other entries, it opens to 0. If $b = 1$, \mathcal{S} chooses a random cycle C and reveals to the adversary only the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C$ (and the adversary checks that all revealed values are 1 and the corresponding entries form a simple n -cycle).

Denote by τ^3, aux^3 the simulated view and the auxiliary information, respectively.

4. In Stage 4, \mathcal{S} commits using the concurrent trapdoor simulator Ts^4 of NMCTCom . It works similarly as in Stage 3. Denote by τ^4, aux^4 the simulated view and the auxiliary information, respectively.
5. In Stage 5, \mathcal{S} receives decommitment information to r from \mathcal{A} and checks the correctness of the opening. If the opening fails, abort the current execution.
6. In Stage 6, \mathcal{S} invokes Ts^2 on inputs (r, aux^2) and gets the decommitment information to r . It also invokes Ts^3 on inputs (r, aux^3) and gets the decommitment information to r . Moreover, it invokes Ts^4 on inputs (r, aux^4) and gets the decommitment information to r . \mathcal{S} transforms all the decommitment information to a witness C (i.e., a Hamiltonian cycle) to the reduced directed graph G . \mathcal{S} executes the MBZKProof protocol using C as witness. Note that \mathcal{S} generates the commitment in MBZKProof using “fake” committing algorithm Com' of AIDCom . For simplicity and without loss of generality, we only present the simulation for Blum’s basic protocol for Hamiltonicity.
 - (a) In the first step, select a random permutation π of the vertices of G , and commit to the adjacency matrix of $\pi(G)$ using the fake algorithm Com' of protocol AIDCom . That is, for each entry of the matrix, compute $\text{Com}'(G; U_{p(n)})$.
 - (b) In the second step, receive a challenge bit b from the adversary \mathcal{A} .
 - (c) In the third step, if $b = 0$, send decommitments to all entries in the adjacency matrix and π . That is, for the $(\pi(u), \pi(v))$ th entry of the matrix, if $(u, v) \in G$, reveal the commitment to 1.

⁵The concurrent trapdoor simulator is designed especially for the proof of CNMZK. Please refer to Section B.3.1 for details.

Otherwise, reveal the commitment to 0. To interpret the $(\pi(u), \pi(v))$ th entry as a commitment to $e \in \{0, 1\}$, run algorithm $\text{Adapt}(G, C, e, c_{\pi(u), \pi(v)}, \rho)$ of AIDCom where $c_{\pi(u), \pi(v)}$ denotes the commitment to the $(\pi(u), \pi(v))$ th entry of the matrix and ρ is the randomness used in the computation of $c_{\pi(u), \pi(v)}$.

If $b = 1$, reveal only the commitments to entries $(\pi(u), \pi(v))$ with $(u, v) \in C$. That is, open the commitment to 1 by running algorithm $\text{Adapt}(G, C, 1, c_{\pi(u), \pi(v)}, \rho)$ of AIDCom .

Extraction of the witnesses: In each right interaction that completes successfully and the verifier is not corrupted, \mathcal{S} extracts a witness w from CECTCom_{sb} committed to by \mathcal{A} in Stage 2.

Handling adaptive corruptions:

Corruption of a verifier: When a verifier \mathcal{V}_i is corrupted, the simulator \mathcal{S} has to fill the view of \mathcal{A} by providing the internal information of \mathcal{V}_i . Since \mathcal{S} simulates \mathcal{V}_i in the right interactions by honestly following the honest verifier strategy and the verifier has no secret information, \mathcal{S} just sends the internal randomness of the simulated \mathcal{V}_i to \mathcal{A} .

Corruption of a prover: When a prover \mathcal{P}_i is corrupted, the simulator \mathcal{S} is then entitled to the prover's input statement x_i and the corresponding witness w_i . \mathcal{S} hands x_i and w_i to \mathcal{A} . In addition, \mathcal{S} has to fill the heretofore unknown portions in the adversary's view. According to the point where \mathcal{P}_i is corrupted, we consider the following cases.

Corruption at the outset of the protocol: Do nothing.

Corruption in Stage 1: The extraction here uses a PRS simulator of [PRS02], an oblivious simulator that is identical to the Killian-Petrank (KP) simulator [KP01]. We follow the analysis of [Ros06]. As the simulator \mathcal{S} acts as an honest receiver of CECom_{sh} in this stage and the messages are independent of the witness w_i , \mathcal{S} only provides \mathcal{A} with the internal randomness of the simulated \mathcal{P}_i .

Corruption in Stage 2: \mathcal{S} proceeds as above for the handling of Stage 1. In addition, \mathcal{S} has to interpret \mathcal{T}_2 as a commitment to the witness w_i whereas \mathcal{T}_2 is a fake commitment. \mathcal{S} runs the concurrent trapdoor simulator of CECTCom_{sb} to open \mathcal{T}_2 to w_i . Finally \mathcal{S} provides \mathcal{A} with the random coins generated above.

Corruption in Stage 3: The handling of Stage 1 and 2 is the same as before. In addition, \mathcal{S} has to interpret \mathcal{T}_3 as a commitment to the witness w_i whereas \mathcal{T}_3 is actually a fake commitment. Note that it does not suffice to invoke the concurrent trapdoor simulator of NMCTCom alone, which provides a simulated decommitment information. Moreover, \mathcal{S} also has to provide the randomness used in the special-sound proof. More formally, \mathcal{S} proceeds as follows:

Corruption in CTCom_{sb} : \mathcal{S} has to invoke the concurrent trapdoor simulator of NMCTCom (which uses the concurrent trapdoor simulator of CTCom_{sb}) to explain the commitment as to w_i . Note that knowing \vec{e}_3 in advance, the construction of this simulator is straightforward. Denote by dec_i the decommitment information to w_i .

Corruption in the \mathcal{WI} proof of knowledge: Since it is public-coin, \mathcal{S} runs the proof by following the honest verifier strategy. \mathcal{S} hands the randomness for the simulated \mathcal{P}_i to \mathcal{A} .

Corruption in special-sound \mathcal{WI} proofs: Recall that for each execution of the special-sound \mathcal{WI} proof, \mathcal{S} uses “fake witness” (i.e., concurrent trapdoor simulator of CTCom_{sb}). Now it has to interpret the proof as one generated using the witness w_i . Without loss of generality, we only describe the strategy of \mathcal{S} for Blum's basic protocol for Hamiltonicity.

Corruption after Step 1 and before Step 3: Recall that Step 1 messages consist of commitments to a randomized permutation of the original graph, and the computation has no relation with the witness. \mathcal{S} now gets the permutation π and runs concurrent trapdoor simulator of CTCom_{sb} to provides \mathcal{A} with the randomness of the simulated \mathcal{P}_i in this step. More precisely, for each edge $(i, j) \in G$, it opens the commitment corresponding to $(\pi(i), \pi(j))$ entry of the adjacency matrix of $\pi(G)$ to 1, and for all other entries, it opens to 0.

Corruption after Step 3: The computation of this step needs the use of the real witness. \mathcal{S} first computes this witness C' based on the witness w_i and the decommitment information dec_i (by Cook-Levin reduction). Then \mathcal{S} explains the proof as one generated using C' as follows. Note that \mathcal{S} is able to equivocate the commitments generated in this step.

If the challenge of the Step 2 is 0, all randomness in this proof has already been revealed.

\mathcal{S} does nothing. If the challenge of Step 2 is 1, according to the design, only decommitments to a Hamiltonian cycle in the permuted graph are revealed. In addition, \mathcal{S} need provide \mathcal{A} with the randomness of the simulated \mathcal{P}_i used in other part of the permuted graph except the Hamiltonian cycle. More precisely, \mathcal{S} chooses a random permutation π^* between the two Hamiltonian cycles C and C' , i.e., $C = \pi^*(C')$. Let $\pi' = \pi^* \circ \pi$, where \circ denotes the composition of permutations.⁶ Let $H = \pi'(G)$. For every edge $(u, v) \in H$ and $(u, v) \notin \pi'(C')$, \mathcal{S} uses concurrent trapdoor simulator of CTCom_{sb} to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 1. For every edge $(u, v) \notin H$, \mathcal{S} uses the concurrent trapdoor simulator CTCom_{sb} to obtain random coins such that the appropriate committed value in the adjacency matrix is a commitment to 0. \mathcal{S} provides \mathcal{A} with the input w_i and the set of random coins described above as well as π' .

Corruption in Stage 4: \mathcal{S} provides the adversary \mathcal{A} with the randomness of the simulated \mathcal{P}_i in Stage 1, 2 and 3 just as above. In addition, \mathcal{S} proceeds as in Stage 3 in the handling of adaptive corruptions in this stage.

Corruption in Stage 5: The handling of Stage 1, 2, 3 and 4 is the same as above. In addition, \mathcal{S} does nothing.

Corruption in Stage 6: The handling of Stage 1, 2, 3, 4 and 5 is the same as above. Recall that \mathcal{S} simulates this stage using a “fake witness”. Upon corruption, \mathcal{S} has to interpret the proof as one generated using real witness w . Without loss of generality, we only describe the strategy of \mathcal{S} for simulation of the Blum’s basic protocol for Hamiltonicity.

Corruption after Step 1 and before Step 3: Recall that \mathcal{S} generates the message in Step 1 using fake committing algorithm Com' of AIDCom . Upon corruption, \mathcal{S} gets the permutation π (selected by itself), and uses the adaptive opening algorithm Adapt of AIDCom to get random coins such that the commitment value of the $(\pi(u), \pi(v))$ th entry in the adjacency matrix of $\pi(G)$ is e , where e equals 0 if $(u, v) \in G$; otherwise e equals 1. \mathcal{S} provides the \mathcal{A} with the input w_i and the set of random coins generated by Adapt as well as with π .

Corruption after Step 3: Denote by C' the new Hamiltonian cycle in G reduced from the witness w_i (in addition to decommitment information of $\mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ to w_i). If $e = 0$, all randomness in this proof has already been revealed. \mathcal{S} does nothing. If $e = 1$, \mathcal{S} has to explain the commitments $\text{Com}'(\pi(G))$ as those generated using the witness C' , i.e., there exists a permutation π' , such that $\pi'(C')$ maps to $\pi(C)$, and the commitments already opened and unopened in $\text{Com}'(\pi(G))$ are consistent to $\pi'(G)$. More precisely, \mathcal{S} proceeds as follows: find a random permutation π^* between the two Hamiltonian cycles C' and C , i.e., $C = \pi^*(C')$. Let $\pi' = \pi^* \circ \pi$ and compute $H = \pi'(G)$. For every edge $(u, v) \in H$ and $(u, v) \notin \pi'(C')$, \mathcal{S} uses algorithm Adapt to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 1. For every edge $(u, v) \notin H$, \mathcal{S} uses algorithm Adapt to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 0. \mathcal{S} provides \mathcal{A} with the input w_i and the set of random coins described above as well as π' .

Post-execution corruption: \mathcal{S} provides the adversary \mathcal{A} with randomness of the simulated \mathcal{P}_i from Stage 1 to Stage 6 just as above.

□

Completing Theorem 1.1 and Theorem 1.2. The CNMZKProof is a $\tilde{\mathcal{O}}(\log n)$ -round adaptively secure CNMZK proof assuming the existence of collision-resistant hash functions. If we replace the commitment CECom_{sh} in Stage 1 with CECom_{sb} , then we get a $\tilde{\mathcal{O}}(\log n)$ -round adaptively secure CNMZK argument. Note that the resulting protocol also assumes the existence of collision-resistant hash functions (needed by NMCTCom or CECTCom_{sb}). In order to get a $\text{poly}(n)$ -round adaptively secure CNMZK proof/argument based on one-way functions, we have to replace the statistically hiding commitment used in CECom_{sh} , CECTCom_{sb} and NMCTCom with the commitment from one-way functions by Haitner et al. [HNO⁺09].

⁶The notation $\pi^* \circ \pi$ means that to apply π^* first and then apply π .

References

- [Bea96] Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *STOC*, pages 629–638, 1996. 2
- [Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986. 7, 9, 21, 36
- [BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354. IEEE Computer Society, 2006. 2
- [CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001. 2
- [CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002. 2
- [CO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On concurrent zero-knowledge with pre-processing. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 485–502. Springer, 1999. 3, 5, 16, 18
- [DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. 1, 6, 21
- [DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, 2002. 2
- [DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998. 1
- [DPP97] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology*, 10(3):163–194, 1997. 15
- [FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer, 1989. 17, 22
- [FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426. ACM, 1990. 22, 24
- [GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996. 1, 15
- [GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304. ACM, 1985. 1
- [Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1*. Cambridge University Press, UK, 2001. 15
- [Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 695–704. ACM, 2011. 3
- [HNO⁺09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009. 12, 15, 16
- [IOS97] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *J. Cryptology*, 10(1):37–50, 1997. 8, 17

- [Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128. Springer, 2007. 2
- [KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *STOC*, pages 560–569, 2001. 11
- [Lin09] Yehuda Lindell. Adaptively secure two-party computation with erasures. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 117–132. Springer, 2009. 2
- [LP09] Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *STOC*, pages 189–198. ACM, 2009. 6, 18
- [LP11a] Huijia Lin and Rafael Pass. Concurrent non-malleable zero knowledge with adaptive inputs. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 274–292. Springer, 2011. 2
- [LP11b] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 705–714. ACM, 2011. 3
- [LPTV10] Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkatasubramanian. Concurrent non-malleable zero knowledge proofs. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 429–446. Springer, 2010. 2, 8, 16, 26, 29
- [LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkatasubramanian. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 571–588. Springer, 2008. 3, 7, 8, 17, 21, 22, 23
- [LZ09] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 183–201. Springer, 2009. Full version, Cryptology ePrint Archive, Report 2009/366. 2, 4, 8, 9, 17, 30
- [MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006. 5, 6, 16, 19, 20, 25
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. 15, 21
- [OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2010. 2
- [PR08] Rafael Pass and Alon Rosen. Concurrent nonmalleable commitments. *SIAM J. Comput.*, 37(6):1891–1925, 2008. 22
- [PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375. IEEE Computer Society, 2002. 5, 6, 11, 16, 20
- [PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2009. 16
- [Ros06] Alon Rosen. *Concurrent Zero-Knowledge - With Additional Background by Oded Goldreich*. Information Security and Cryptography. Springer, 2006. 11

A Basic Definitions

We recall the definition of a witness relation for an NP language.

Definition A.1 (Witness Relation). A witness relation for a language $L \in \text{NP}$ is a binary relation R_L that is polynomial bounded, polynomial-time recognizable and characterizes by $L = \{x \mid \exists w \text{ s.t. } (x, w) \in R_L\}$.

We say that y is a witness for the membership of x in L if $(x, y) \in R_L$. We let $R_L(x)$ denote the set of witnesses for the membership of x in L , i.e., $R_L(x) = \{y : (x, y) \in L\}$. In the following, we assume a fixed witness relation R_L for each language $L \in \text{NP}$.

A function $\mu(\cdot)$, where $\mu : \mathbb{N} \rightarrow [0, 1]$ is called *negligible* if for every positive polynomial $p(\cdot)$, for all sufficiently large $n \in \mathbb{N}$, $\mu(n) < \frac{1}{p(n)}$. A *probability ensemble* is a sequence $X = \{X_s\}_{s \in S}$ of random variables, where S is a set of strings and X_s is a random variable ranging over $\{0, 1\}^{p(|s|)}$ for some polynomial $p(\cdot)$. We assume the reader is familiar with commitment schemes and interactive proofs.

Definition A.2 (Computational Indistinguishability). Let S be a set of strings. We say that the two probability ensembles $X = \{X_s\}_{s \in S}$ and $Y = \{Y_s\}_{s \in S}$ are *computationally indistinguishable*, denoted by $X \stackrel{c}{\equiv} Y$, if for every probabilistic polynomial-time (PPT) distinguisher D , every polynomial $p(\cdot)$, all sufficiently long $s \in S$ and all auxiliary information $z \in \{0, 1\}^{\text{poly}(n)}$,

$$\Pr[D(X_s, s, z)] - \Pr[D(Y_s, s, z)] < \frac{1}{p(|s|)}$$

Definition A.3 (Statistical Closeness). Let $S \subseteq \{0, 1\}^*$ be a set of strings. We say that the two probability ensembles $X = \{X_s\}_{s \in S}$ and $Y = \{Y_s\}_{s \in S}$ are *statistically close* or *statistically indistinguishable*, denoted by $X \stackrel{s}{\equiv} Y$, if for all sufficiently long $s \in S$, the statistical distance $\sum_{\alpha} |\Pr[X_s = \alpha] - \Pr[Y_s = \alpha]|$ is negligible.

A.1 Commitment Schemes

A commitment scheme is a basic cryptographic primitive which is usually seen as a digital analogue of the sealed envelope. It is a two-phase protocol between a committer and a receiver. In the commit phase, the committer puts its message in a box, locks the box and hands it to the receiver. Receiving the box, the receiver does not know the exact message in the box. This is called the hiding property. In the reveal phase, the committer gives the key to the receiver. The receiver then opens the box and retrieves the message. The message should be the same as the one chosen by the committer. This is called the binding property. Commitment schemes come in two different flavors, statistically hiding and statistically binding. We only sketch the properties of both the flavors.

Statistically binding: In a statistically binding commitment, the binding property holds against unbounded adversaries, i.e., even an all-powerful adversary is not able to generate a commitment that later is opened to two different values. The hiding property holds against computationally bounded adversaries, i.e., commitments to two different values are computationally indistinguishable.

Statistically hiding: In a statistically hiding commitment scheme, the hiding property holds against unbounded adversaries, i.e., commitments to any two different values are statistically close. The binding property holds only against computationally bounded adversaries, i.e., no polynomial-time adversary is able to open a commitment in two different ways.

Non-interactive statistically binding commitment can be constructed from any 1-1 one-way functions [Gol01]. Based on one-way functions, there exists two-round statistically binding commitment [Nao91]. Statistically hiding commitment scheme can be constructed from any one-way functions [HNO⁺09]. However, constant-round ones are known to exist under the stronger assumptions, such as the certified claw-free permutations [GK96], and collision-resistant hash functions [DPP97].

A.2 Concurrently Extractable Commitment Schemes

The notion of concurrently extractable commitment scheme is first introduced by Micciancio et al. [MOSV06] and is implicit in [PRS02]. This is an abstraction of the preamble stage of the concurrent zero-knowledge protocol of [PRS02]. Roughly, a commitment scheme is concurrently extractable if there exists an efficient extractor that is able to generate a view that is statistically indistinguishable with the view of a malicious committer in the commit phases, and moreover, extract the committed values from any valid commitment sent by the committer.

Definition A.4 (Concurrently Extractable Commitment Scheme). Let $\langle C, R \rangle$ be a statistically hiding (resp. statistically binding) commitment scheme. We say that $\langle C, R \rangle$ is a **concurrently extractable commitment scheme** if there exists an (expected) PPT oracle machine (the extractor) \mathcal{E} such that for every polynomial $p = p(n)$, for every any unbounded (resp. PPT) p -concurrent adversarial committer C^* , outputs a pair $\vec{\tau}, \mathbb{V}$ such that the following properties hold:

Simulation: $\vec{\tau}$ is identically distributed to the view of C^* with an honest receiver R in all commit phases.

Concurrent extraction: Denote by $\vec{\tau} = (\tau_1, \dots, \tau_p)$, where $p = p(n)$ is a polynomial and τ_i is the view of the committer C^* in the i th execution. Denote by $\mathbb{V} = (v_1, \dots, v_p)$. The probability that there exists $i \in [p]$ such that τ_i is accepting and $v_i = \perp$ is negligible.

We say that a commitment scheme is concurrently extractable and trapdoor if it is a concurrently extractable commitment and a trapdoor commitment. Moreover, we say that a commitment is concurrently extractable and concurrent trapdoor if it is a concurrently extractable commitment and a concurrent trapdoor commitment.

Assuming the existence of collision-resistant hash functions, there exists $\tilde{\mathcal{O}}(\log n)$ -round concurrently extractable statistically hiding commitment schemes [PRS02, MOSV06]. If we only assume the existence of one-way functions, the round complexity changes to $\text{poly}(n)$ [HNO⁺09]. Assume the existence of one-way functions, there exists $\tilde{\mathcal{O}}(\log n)$ -round concurrently extractable statistically binding commitment schemes [LPTV10]. We emphasize here that these constructions all have a non-interactive reveal phase.

A.3 Concurrent Trapdoor Commitment Schemes

Roughly, a trapdoor commitment is a commitment scheme with an additional property such that there exists a simulator, with knowledge of some trapdoor information, can overcome the binding property and open a commitment arbitrarily. We extend this notion to concurrent execution setting and define concurrent trapdoor commitment schemes. Let $p = p(n)$ be a polynomial. We say that R^* is a p -concurrent malicious receiver if it performs at most p concurrent executions with a committer. We use the notation $\{\cdot\}_{\mathbb{V}, n, z}$ as shorthand for $\{\cdot\}_{\mathbb{V} \in \{0,1\}^{n \cdot p}, n \in \mathbb{N}, z \in \{0,1\}^*}$.

Definition A.5 (Concurrent Trapdoor Commitment Scheme). Let $\langle C, R \rangle$ be a statistically binding (resp. statistically hiding) commitment scheme. We say that $\langle C, R \rangle$ is a **concurrent trapdoor commitment scheme** if there exists an (expected) PPT oracle machine (i.e., concurrent trapdoor simulator) $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ such that for any polynomial $p = p(n)$, for any PPT p -concurrent malicious receiver R^* and for all $\mathbb{V} = (v_1, \dots, v_p), v_1, \dots, v_p \in \{0,1\}^n$, the following two probability distributions are computationally indistinguishable (resp. statistically indistinguishable):

- $\{\text{sta}_{\langle C, R \rangle}^{R^*}(1^n, \mathbb{V}, z)\}_{\mathbb{V}, n, z}$, where $\text{sta}_{\langle C, R \rangle}^{R^*}(1^n, \mathbb{V})$ denotes the random variable describing the receiver's view of the interactions in the commit phases and reveal phases with $C(1^n, \mathbb{V})$.
- $\left\{(\vec{\tau}, \text{aux}) \leftarrow \text{Ts}_1^{R^*}(1^n, z), \vec{\omega} \leftarrow \text{Ts}_2(\text{aux}, \mathbb{V}) : (\vec{\tau}, \vec{\omega})\right\}_{\mathbb{V}, n, z}$, where $\vec{\tau} = (\tau_1, \dots, \tau_p)$ and $\vec{\omega} = (\omega_1, \dots, \omega_p)$.

Note that τ_i is the simulated view for the i th commit phase, and ω_i is the decommitment information for the i th reveal phase.

Remark 3. In the above definition, we only require the concurrent trapdoor simulator to simulate the view of a PPT (not unbounded) adversary R^* interacting with an honest committer.

Considering statistically hiding commitments, D. Crescenzo et al. [CO99] constructed a constant-round trapdoor commitment scheme based on the hardness of discrete-logarithm problem. Pass and Wee [PW09] gave a black-box construction of trapdoor commitment scheme from any one-way functions.

Their protocol is only computationally hiding and computationally binding. Considering statistically binding commitments, we will give constructions of trapdoor commitment schemes in this paper. Previous work only use computationally binding trapdoor commitments. We find that the statistically binding ones are also very useful in achieving adaptive security of zero-knowledge protocols. We believe this variant to be of independent interest.

A.4 Adaptive Instance-Dependent Commitment Schemes

Instance-dependent commitment schemes were first introduced in [IOS97]. Roughly speaking, an instance-dependent commitment scheme is a commitment whose properties depend on whether the instance in question is in the language or not. Typically, it is defined for a language L as follows. Let x be a statement. If $x \in L$ then the commitment associated with x is computationally hiding, and if $x \notin L$ then the commitment associated with x is statistically binding. The adaptive instance-dependent commitment scheme was first introduced by Lindell and Zarusim [LZ09]. Roughly, it has the additional property that commitments are equivocal. It has a fake algorithm which generates fake commitments. If knowing the witness to the statement, the fake commitment can be opened to arbitrary value; otherwise, it cannot necessarily be opened to any value.

Definition A.6 (Adaptive Instance-Dependent Commitment Schemes). Let R be an NP relation and let L be an NP language associated with R . $(\text{Com}, \text{Com}', \text{Adapt})$ is an adaptive instance-dependent commitment scheme for L if the following conditions hold:

- **Efficiency:** $\text{Com}, \text{Com}', \text{Adapt}$ are all probabilistic polynomial-time algorithms.
- **Computational Hiding:** For every $x \in L$, the following holds:

$$\{\text{Com}(x, 0)\}_{x \in L} \stackrel{c}{\equiv} \{\text{Com}(x, 1)\}_{x \in L} \stackrel{c}{\equiv} \{\text{Com}'(x)\}_{x \in L}$$

- **Statistical Binding:** For every $x \notin L$, there exists a negligible function $\varepsilon(\cdot)$, such that $\Pr[\text{Com}(x, 0; r) = \text{Com}(x, 1; r')] < \varepsilon(|x|)$.
- **Adaptive Trapdoor:** For all $x \in L$, all $c' = \text{Com}'(x; U_{p(|x|)})$ and all bit $b \in \{0, 1\}$, and for all $w \in R(x)$ the following holds: $\{c', \text{Adapt}(x, w, b, c', U_{p(|x|)})\}_{x \in L, w \in R(x)} \stackrel{c}{\equiv} \{\text{Com}(x, b; U_{p(|x|)}), b, U_{p(|x|)}\}_{x \in L, w \in R(x)}$.

Note that Com is an ordinary committing algorithm. Com' is a “fake” committing algorithm. Adapt is an adaptive opening algorithm. When $x \in L$, Com' creates commitments that are not associated to any specific value. However, given a witness $w \in R(x)$ to the fact $x \in L$, the algorithm Adapt can explain every output c' of Com' as a valid commitment to any bit b . But without such a witness, a commitment generated by Com' can not necessarily to be decommitted to any bit.

Lindell and Zarusim [LZ09] constructed an adaptive instance-dependent commitment scheme by slightly modifying the trapdoor commitment scheme of [FS89]. Their scheme is only for a single bit. By running their atomic scheme in parallel, they obtained an adaptive instance-dependent commitment scheme for multiple bits. The reader is referred to the full version of [LZ09] for the explicit constructions and the formal security definition of adaptive trapdoor property for instance-dependent *string* commitment scheme.

A.5 Non-Malleable Commitment Schemes

We recall the definition of non-malleability from [LPV08]. Let $\langle C, R \rangle$ be a tag-based commitment scheme. Consider a man-in-the-middle adversary \mathcal{A} that participates in one left interaction and one right interaction simultaneously. In the left interaction, \mathcal{A} interacts with an honest committer under tag id of its choice. \mathcal{A} will receive a commitment to a value v . In the right interaction, \mathcal{A} tries to commit to a related value \tilde{v} under tag $\tilde{\text{id}}$ of its choice. If the right commitment fails, or undefined, or $\text{id} = \tilde{\text{id}}$, its value is set to \perp . Let $\text{nmc}_{\langle C, R \rangle}^{\mathcal{A}}(v, z)$ be a random variable that describe the value \tilde{v} combined with the view of \mathcal{A} in the above experiment. Given a function $t = t(n)$, we use notation $\{\dots\}_{n, v, v', z, \text{id}}$ as shorthand for $\{\dots\}_{n \in \mathbb{N}, v \in \{0, 1\}^n, v' \in \{0, 1\}^n, z \in \{0, 1\}^*, \text{id} \in \{0, 1\}^t}$.

Definition A.7 (Non-Malleable Commitments). Let $\langle C, R \rangle$ be a statistically binding commitment. We say that $\langle C, R \rangle$ is non-malleable (with respect to itself) with tags of length $t = t(n)$ if

for every PPT man-in-the-middle adversary \mathcal{A} , the two ensembles $\{\text{nmc}_{\langle C, R \rangle}^{\mathcal{A}}(v, z, \text{id})\}_{n, v, v', z, \text{id}}$ and $\{\text{nmc}_{\langle C, R \rangle}^{\mathcal{A}}(v', z, \text{id})\}_{n, v, v', z, \text{id}}$ are computationally indistinguishable.

We say that a commitment scheme $\langle C, R \rangle$ is non-malleable (concurrent) trapdoor commitment scheme if it is non-malleable commitment scheme and a (concurrent) trapdoor commitment scheme.

Non-malleable commitment robust w.r.t k -round protocols. The notion of non-malleability w.r.t k -round protocols was first introduced by Lin and Pass [LP09]. Previously non-malleability w.r.t commitment only considers man-in-the-middle adversaries that participates executions of the same protocols on the left and on the right. Robust non-malleability, on the other hand, considers man-in-the-middle adversaries that participates in executions of arbitrary protocol on the left. We recall the definition from [LP09]. The man-in-the-middle adversary \mathcal{A} interacts with \mathcal{B} on the left, while simultaneously acting as a committer using a commitment scheme $\langle C, R \rangle$. The input to \mathcal{B} is y . On the right, \mathcal{A} commits to \tilde{v} using tag of its choice. We let $\text{nmc}_{\langle C, R \rangle}^{\mathcal{B}, \mathcal{A}}(y, z)$ be the random variable that describes the value \tilde{v} combined with the view of \mathcal{A} in the above experiment. Intuitively, we say that $\langle C, R \rangle$ is non-malleable w.r.t \mathcal{B} if $\text{nmc}_{\langle C, R \rangle}^{\mathcal{B}, \mathcal{A}}(y_1, z)$ and $\text{nmc}_{\langle C, R \rangle}^{\mathcal{B}, \mathcal{A}}(y_2, z)$ are indistinguishable whenever interactions with $\mathcal{B}(y_1)$ and $\mathcal{B}(y_2)$ are indistinguishable.

Definition A.8 (Non-malleability w.r.t \mathcal{B}). Let $\langle C, R \rangle$ be a statistically binding commitment scheme. Let \mathcal{B} be a PPT machine. We say the commitment scheme $\langle C, R \rangle$ is non-malleable w.r.t \mathcal{B} if for every two sequences $\{y_n^1\}_{n \in \mathbb{N}}$ and $\{y_n^2\}_{n \in \mathbb{N}}$, such that for all PPT machine \tilde{A} , it holds that

$$\left\{ \langle \mathcal{B}(y_n^1), \tilde{A}(z) \rangle(1^n) \right\}_{n \in \mathbb{N}, z \in \{0, 1\}^*} \stackrel{c}{\equiv} \left\{ \langle \mathcal{B}(y_n^2), \tilde{A}(z) \rangle(1^n) \right\}_{n \in \mathbb{N}, z \in \{0, 1\}^*}$$

where $\langle \mathcal{B}(y), \tilde{A}(z) \rangle(1^n)$ denotes the view of \tilde{A} after interaction with \mathcal{B} on common input 1^n , and private inputs y and z respectively, then it also holds that, for every PPT man-in-the-middle adversary \mathcal{A}

$$\left\{ \text{nmc}_{\langle C, R \rangle}^{\mathcal{B}, \mathcal{A}}(y_n^1, z) \right\}_{n \in \mathbb{N}, z \in \{0, 1\}^*} \stackrel{c}{\equiv} \left\{ \text{nmc}_{\langle C, R \rangle}^{\mathcal{B}, \mathcal{A}}(y_n^2, z) \right\}_{n \in \mathbb{N}, z \in \{0, 1\}^*}$$

We say that $\langle C, R \rangle$ is non-malleable w.r.t k -round protocols if $\langle C, R \rangle$ is non-malleable w.r.t any PPT machine \mathcal{B} that interacts with the man-in-the-middle adversary in k rounds. Below, we focus on commitment schemes that are non-malleable w.r.t itself and arbitrary $\ell(n)$ -round protocols, where ℓ is a super-logarithmic function. We say that such a commitment scheme is robust w.r.t $\ell(n)$ -round protocols.

B Proofs of the Commitment Schemes

B.1 Concurrent Trapdoor Commitments

Before proving the protocol CTCom_{sb} in Figure 1 is a concurrent trapdoor statistically binding commitment scheme, we emphasize that Naor's commitment scheme is equivocal if the first message is generated by a coin-tossing protocol between the committer and the receiver. The reader is referred to [CO99] for details. Here for self-reference, we give a brief introduction to the scheme.

Claim B.1. *The protocol CTCom_{sb} in Figure 1 is a concurrent trapdoor statistically binding commitment scheme.*

Proof. We need to prove the protocol CTCom_{sb} satisfies the following three properties: computational hiding, statistical binding and concurrent trapdoor.

Computational hiding. The hiding property follows from the hiding property of Com_{sb} . By using standard hybrid argument, we can show that the protocol CTCom_{sb} is computationally hiding. More formally, fix a cheating receiver R^* , two n -bit values v_1, v_2 and suppose we want to show that $\text{CTCom}_{sb}(v_1)$ and $\text{CTCom}_{sb}(v_2)$ are indistinguishable. We define a sequence of hybrids. For each $0 \leq i \leq n$, in the i th hybrid, the first i bits are from v_1 and the last $n - i$ bits are from v_2 . It is straightforward that the zeroth hybrid is identical to $\text{CTCom}_{sb}(v_2)$, and the n th hybrid is identical to $\text{CTCom}_{sb}(v_1)$. Next by the hiding property of Com_{sb} we get that the transcript of the $(i - 1)$ th hybrid and i th hybrid are indistinguishable. By a union bound, we conclude that $\text{CTCom}_{sb}(v_1)$ and $\text{CTCom}_{sb}(v_2)$ are indistinguishable.

Statistical binding. Roughly, as the commitment of CECom_{sh} is statistically hiding, even an infinitely powerful adversary cannot guess the challenge of the receiver. Thus, it cannot program the coin-tossing result to one which allows equivocation. With all but negligible probability the result r of the coin-tossing is uniformly random. Then following from the property of Naor’s commitment scheme, we get the commitment is statistically binding.

Concurrent trapdoor: We construct a PPT concurrent trapdoor simulator $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ such that for any polynomial $p = p(n)$, for any PPT p -concurrent malicious receiver R^* and for all $\mathbb{V} = (v_1, \dots, v_p)$ where $v_1, \dots, v_p \in \{0, 1\}^n$, the view of R^* in real executions and the output of Ts (i.e., the simulated transcript) are computationally indistinguishable. More formally, on input the security parameter $1^n, z$, $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ proceeds as follows.

1. Ts_1 interacts with the p -concurrent malicious receiver $R^*(z)$.
2. Ts_1 acts as an honest committer and waits commitments from R^* (under the CECom_{sh}). Ts_1 proceeds as follows: Ts_1 invokes the concurrent extractor of CECom_{sh} . Denote by $\vec{\tau} = (\tau_1, \dots, \tau_p)$ the simulated views and $\vec{r}' = (r'_1, \dots, r'_p)$ the extracted challenges.
3. Ts_1 next prepares the result of the coin-tossing. For the i th interaction, if $r'_i = \perp$, then Ts_1 outputs the transcript up to now and aborts the i th commitment. Otherwise, it chooses a random string r''_i such that $r'_i \oplus r''_i$ allows equivocation in the following execution. Ts_1 records the trapdoor in aux_i . Next if the adversary R^* opens its commitment to a value different from r'_i , then Ts_1 outputs the transcript up to now and aborts the i th commitment.
4. Ts_1 next prepares the commitments to \mathbb{V} . For each $i \in [p]$, if the i th commitment is not aborted, Ts_1 generates a “fake” commitment to v_i using Com_{sb} . Ts_1 records the transcript in the i th commit phase in τ_i .
5. In the i th reveal phase, to open to v_i , $\text{Ts}_2(\tau_i, \text{aux}_i)$ generates the decommitment information of Com_{sb} using aux_i . Ts_2 records the decommitment information in ω_i .
6. Output $\vec{\tau} = (\tau_1, \dots, \tau_p), \vec{\omega} = (\omega_1, \dots, \omega_p)$.

Next we show that the view of R^* with an honest committer and the simulated transcript $(\vec{\tau}, \vec{\omega})$ are indistinguishable. Note that if the extraction of CECom_{sh} succeeds and for each commit phase R^* opens to a challenge the same as the one extracted (i.e., Ts_1 does not abort), then the view of R^* with an honest committer and the simulated transcripts generated by Ts are computationally indistinguishable following from the hiding property of the Com_{sb} . Due to the concurrent extractability property of protocol CECom_{sh} , the extraction fails only with negligible probability for valid commitments. Moreover, since the commitment CECom_{sh} is computationally binding, R^* is able to open its commitment to a challenge different from the one extracted only with negligible probability. Thus, we conclude that the view of R^* in real executions is indistinguishable from $\vec{\tau}, \vec{\omega}$. \square

B.2 Concurrently Extractable and Concurrent Trapdoor Commitments

In this section, we give a construction of concurrently extractable and concurrent trapdoor commitment. Let CTCom_{sb} be a statistically binding concurrent trapdoor commitment scheme. Let $\ell = \ell(n)$ be any super logarithmic function. The protocol is shown in Figure 2.

Claim B.2. *The protocol CECTCom_{sb} in Figure 2 is a concurrently extractable and concurrent trapdoor statistically binding commitment scheme.*

Proof. We need to prove the protocol CECTCom_{sb} satisfies the following four properties: computational hiding, statistical binding, concurrent extractability and concurrent trapdoor. Note that the commitment used by the committer is replaced with a concurrent trapdoor one compared with the committer in protocol CECom_{sb} . The first three properties are not affected by this change. Thus, the proof of the first three properties are virtually identical to that in [MOSV06]. We only give a proof sketch for these three properties.

Computational hiding. The hiding property follows from the hiding property of CTCom_{sb} . We give a hybrid argument. Suppose, on the contrary, there exists a PPT distinguisher D and a polynomial $p(n)$ such that for infinitely many $n \in \mathbb{N}$, there exists strings $v_1, v_2, |v_1| = |v_2| = n$, such that D distinguishes

between the probability ensembles $\text{CECTCom}_{sb}(v_1)$ and $\text{CECTCom}_{sb}(v_2)$ with probability at least $\frac{1}{p(n)}$. We then define the following hybrid committers H_i . H_i runs as an honest committer C . It commits to independent random shares of v_1 for the first i pairs, and independent random shares of v_2 for the last $n - i$ pairs. It is straightforward that the hybrid H_0 is the same as $\text{CECTCom}_{sb}(v_2)$ and the hybrid H_n is identical to $\text{CECTCom}_{sb}(v_1)$. By standard hybrid argument, there exists an $i \in [n]$ such that D distinguishes the view of a receiver in the hybrids H_{i-1} and H_i with probability at least $\frac{1}{np(n)}$. Recall that the only difference between the hybrids H_{i-1} and H_i is that in the former, the i th pair is independent random shares of v_2 whereas in the latter it is independent random shares of v_1 . Since both of the hybrids only reveal one of their random shares, we break the hiding property of CTCom_{sb} .

Statistical binding. Note that the value v (in addition to many shares) is committed to using CTCom_{sb} in the commit phase and opened in the reveal phase. The binding property follows directly from the binding property of CTCom_{sb} . As CTCom_{sb} is statistically binding, the protocol CTCom_{sb} is also statistically binding.

Concurrent extractability. Recall that the difference between the commitment CECom_{sb} and the commitment CECTCom_{sb} is that in the former the committer uses Com_{sb} to commit to v and all $2n\ell$ strings whereas in the latter the committer uses CTCom_{sb} . As the trapdoor property of CTCom_{sb} does not affect the extraction, we can show that CECTCom_{sb} is concurrently extractable following from a similar analysis as in [PRS02, MOSV06].

Concurrent trapdoor. The trapdoor property follows from the trapdoor property of CTCom_{sb} . Next we construct a PPT concurrent trapdoor simulator $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ such that for any polynomial $p = p(n)$, for any PPT p -concurrent adversary R^* and for all $\mathbb{V} = (v_1, \dots, v_p)$, the view of R^* in real executions and the output of Ts (i.e., the simulated transcript) are computationally indistinguishable. More formally, on input the security parameter $1^n, z$, Ts proceeds as follows.

1. Ts_1 interacts with the p -concurrent adversary $R^*(z)$.
2. Ts_1 invokes the concurrent trapdoor $\widetilde{\text{Ts}} = (\widetilde{\text{Ts}}_1, \widetilde{\text{Ts}}_2)$ of CTCom_{sb} on a p' -bounded adversary, where $p' = p(1 + 2n\ell)$. Denote by $\vec{\tau} = (\tau_1, \dots, \tau_{p'})$ and $\text{aux} = (\text{aux}_1, \dots, \text{aux}_{p'})$ the outputs of $\widetilde{\text{Ts}}_1$. Let $\text{tran}_i = (\tau_{(i-1)(1+2n\ell)+1}, \dots, \tau_{i(1+2n\ell)})$ and $\text{info}_i = (\text{aux}_{(i-1)(1+2n\ell)+1}, \dots, \text{aux}_{i(1+2n\ell)})$. Note that tran_i is the simulated view in the i th commitment and info_i is the auxiliary information for the i th commitment.
3. Next, Ts_1 simulates the ℓ rounds of interactions in the i th commitment, where $i \in [p]$. More formally, for each $j \in [\ell]$, Ts_1 proceeds as follows:
 - (a) Ts_1 receives a challenge e_j from R^* .
 - (b) For each $k \in [n]$, choose a random string $\alpha_{k,j}^{e_{k,j}} \in \{0, 1\}^n$. This corresponds to the $(i-1)(1 + 2n\ell) + 1 + 2(j-1)n + 2(k-1) + (1 + e_{k,j})$ th commitment of $\widetilde{\text{Ts}}$.
 - (c) Invoke $\widetilde{\text{Ts}}_2((\alpha_{1,j}^{e_{1,j}}, \dots, \alpha_{n,j}^{e_{n,j}}), \text{info}_i)$ and get outputs $\omega_j = (\omega_{1,j}, \dots, \omega_{n,j})$. Update $\text{tran}_i = \text{tran}_i || e_j || \omega_j$ and $\text{info}_i = \text{info}_i || \omega_j$.
 - (d) Send to R^* the decommitment information ω_j .
4. Set $\text{info} = (\text{info}_1, \dots, \text{info}_p)$.
5. After all the commit phases complete, Ts_2 starts the simulation of the reveal phases. On inputs $\mathbb{V} = (v_1, \dots, v_p), \text{info}$, for the i th reveal phase, Ts_2 proceeds as follows:
 - (a) Extract from info_i all the opened strings $\{\alpha_{k,j}^{e_{k,j}}\}_{k \in [n], j \in [\ell]}$ in the i th commit phase.
 - (b) For each $k \in [n], j \in [\ell]$, compute $\alpha_{k,j}^{1-e_{k,j}} = v_i \oplus \alpha_{k,j}^{e_{k,j}}$. Note that $\alpha_{k,j}^{1-e_{k,j}}$ is the opened value of $(i-1)(1 + 2n\ell) + 1 + 2(j-1)n + 2(k-1) + (1 + 1 - e_{k,j})$ th commitment of $\widetilde{\text{Ts}}$, and v_i is the opened value of $(i-1)(1 + 2n\ell) + 1$ th commitment of $\widetilde{\text{Ts}}$.
 - (c) Invoke $\widetilde{\text{Ts}}_2(v_i, \alpha_{1,1}^{1-e_{1,1}}, \dots, \alpha_{n,\ell}^{1-e_{n,\ell}}, \text{info}_i)$ and gets $\omega_i = (\omega_i^*, \omega_{1,1}, \dots, \omega_{n,\ell})$.
 - (d) Send R^* the decommitment information ω_i .
 - (e) Update $\text{tran}_i = \text{tran}_i || \omega_i$.
6. Output $\text{tran} = (\text{tran}_1, \dots, \text{tran}_p)$.

The outputs of $\text{Ts}(1^n, \mathbb{V}, z)$ and $\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ are indistinguishable following from the concurrent trapdoor property of CTCom_{sb} . To prove this, we design a sequence of hybrid simulators $\{H_i\}_{0 \leq i \leq p}$. Hybrids H_i receives inputs of the values $\mathbb{V} = (v_1, \dots, v_p)$, and proceeds identically as Ts except that for the last $n - i$ commitments, it generates the commitments by following the honest committer strategy. It directly follows from the construction that $H_0(1^n, \mathbb{V}, z) = \text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ and $H_p(1^n, \mathbb{V}, z) = \text{Ts}(1^n, \mathbb{V}, z)$. Next we show that the outputs of hybrids H_i and H_{i+1} are indistinguishable. Note that the difference between the execution of H_i and H_{i+1} is that in the former the i th commitment is generated honestly, whereas in the latter, it is generated by the trapdoor simulator of CTCom_{sb} . It follows from the concurrent trapdoor property of CTCom_{sb} that $H_i(1^n, \mathbb{V}, z) \stackrel{c}{\equiv} H_{i+1}(1^n, \mathbb{V}, z)$. \square

B.3 Non-Malleable Concurrent Trapdoor Commitments

In this section, we give a construction of non-malleable concurrent trapdoor commitment scheme. The protocol is shown in Figure 3. The protocol is based on the commitment in [LPV08] and adapted to our setting.

Before presenting the description of the protocol, we give a short introduction of a special-sound witness-indistinguishable proof system and the message scheduling technique introduced by [DDN00].

Special-sound proofs. A k -round interactive proof for the language $L \in \text{NP}$ with witness relation R_L is special-sound with respect to R_L if the following holds: there exists a deterministic polynomial-time procedure that can extract a witness with overwhelming probability given a randomly sampled $(k - 2)$ -message prefix $\vec{\alpha}$ of the protocol and two independent accepting completions of the prefix $(\vec{\alpha}, \beta, \gamma)$ and $(\vec{\alpha}, \beta', \gamma')$. Special-sound \mathcal{WI} proofs for NP languages can be based on the existence of non-interactive commitment schemes. Assuming only one-way functions, 4-round special-sound \mathcal{WI} proofs for NP languages exist. More precisely, there is a 3-round special-sound \mathcal{WI} proof for the language of Hamiltonian Graphs [Blu86], assuming one-way permutation families exist. If the commitment scheme used by the protocol [Blu86] is replaced by Naor's commitment scheme [Nao91], then it becomes a 4-round special-sound \mathcal{WI} proof while the assumption is reduced to the existence of one-way functions. Moreover, if the commitment scheme used by the protocol [Blu86] is a CTCom_{sb} , then it becomes a k -round special-sound \mathcal{WI} proofs, where k depends on the round complexity of CTCom_{sb} .

1. In Stage 1, the committer commits to v using a concurrent trapdoor commitment CTCom_{sb} . Let com be the commitment.
2. In Stage 2, the receiver picks two random n -bit strings r_1, r_2 and sends its image $s_1 = f(r_1)$ and $s_2 = f(r_2)$. Then it uses a \mathcal{WI} proof of knowledge that there exists a value r such that $s_1 = f(r)$ or $s_2 = f(r)$.
3. In Stage 3, the committer proves that com is either a valid commitment to v , or it knows the preimage of s_0 or s_1 . This is proved by $4t$ invocations of a special-sound \mathcal{WI} proof that the messages are scheduled based on the tag id . More precisely, there are t rounds, where in the i th round, the schedule $\text{design}_{\text{id}_j}$ is followed by $\text{design}_{1-\text{id}_j}$.

Message scheduling technique. The commitment scheme in Figure 3 relies on a special-sound \mathcal{WI} scheduling which encodes the tag of the commitment. The scheduling is shown in Figure 4. This scheduling is vital in achieving the non-malleability. The main advantage of this scheduling is that for the proof given by a man-in-the-middle adversary in the right interaction, there exists a point at which the adversary cannot answer the challenge from the verifier by simply modifying the proof on the left interaction. Note that we extend the message scheduling technique in [LPV08] to cover the k -round special-sound \mathcal{WI} proofs instead of 3-round ones.

Remark 4. There are several differences between our protocol and the commitment scheme in [LPV08]. First, we exchange the sequence of Stage 1 and Stage 2. In the following, when we design an adaptively secure CNMZK protocol, we require the Stage 1 to be combined with other part of the zero-knowledge protocol (See Remark 1). Second, the committer in Stage 1 and Stage 3 uses a concurrent trapdoor commitment scheme CTCom_{sb} instead of a statistically binding commitment in [LPV08]. Third, in order to prove the protocol is hiding, we require the receiver to prove knowledge of either preimage of two

values under one-way function f using a \mathcal{WI} proof of knowledge system. This two-pair technique is well-known in the design of zero-knowledge protocols [FS89].

Remark 5. In the design of CNMZK protocols, we merge the CECom_{sh} part in CTCom_{sb} with other parts of CNMZK, i.e., the receiver runs all of CECom_{sh} first before the execution of CNMCTCom . This means that part of execution in Stage 1 and Stage 3 will be executed in parallel ahead of schedule. Denote by $\text{CNMCTCom}'$ this modified version. We emphasize here that this modification will not affect the property of $\text{CNMCTCom}'$. We will give the rough idea after the proof of CNMCTCom .

Claim B.3. *The protocol CNMCTCom in Figure 3 is a concurrent non-malleable concurrent trapdoor commitment scheme.*

Proof. We need to show that the protocol satisfies the following four properties: computational hiding, statistical binding, concurrent non-malleability and concurrent trapdoor.

Computational hiding: The hiding property essentially follows from the hiding property of CTCom_{sb} in Stage 2 and the fact that Stage 3 of the protocol is \mathcal{WI} (note that \mathcal{WI} property is preserved under sequential composition [FS90]). If there exists an adversary R^* that violates the hiding property of CNMCTCom , we design an algorithm R' that breaks the hiding property of CTCom_{sb} . Without loss of generality, we assume R^* is deterministic. R' internally runs a copy of R^* . It forwards messages in Stage 1 to an external committer of CTCom_{sb} . R' runs as an honest verifier in Stage 2. After the completion of Stage 2, R' keeps rewinding the \mathcal{WI} protocol and extracts a “fake witness”. In Stage 3, R' gives \mathcal{WI} proofs using the fake witness. Finally, it outputs whatever R^* outputs. From the \mathcal{WI} property of Stage 3, it follows that R' distinguishes the commitment made using CTCom_{sb} , if R^* distinguishes the commitment made using NMCTCom .

Statistical binding: The binding property follows directly from the binding property of CTCom_{sb} . As CTCom_{sb} is statistically binding, CNMCTCom is also statistically binding.

Concurrent non-malleability. Pass and Rosen [PR08] showed that if a commitment scheme is one-many non-malleable, then it is also concurrent non-malleable. Thus, we only show the scheme is one-many non-malleable. The proof follows the line of [LPV08]. We present a proof sketch and point out the main difference with the work [LPV08]. Before giving the proof, we define a *determining message* of a commitment scheme which is from the committer and determines the value committed in the commitment. For the protocol CNMCTCom , the determining message falls in Stage 1. More precisely, it is the message in the commitment CTCom_{sb} which consists of commitments to v using Com_{sb} , i.e., the last message of CTCom_{sb} . In the following, we sometimes use $\langle C, R \rangle$ to denote the commitment scheme CNMCTCom .

Just as the proof of [LPV08], for every man-in-the-middle adversary \mathcal{A} , we design an expected non-uniform PPT malicious receiver R^* for a variant of the commitment scheme $\langle C, R \rangle$, which is denoted by $\langle \hat{C}, \hat{R} \rangle$. We postpone the description of $\langle \hat{C}, \hat{R} \rangle$ to a later stage. Upon receiving a commitment to v using $\langle \hat{C}, \hat{R} \rangle$, R^* outputs what is indistinguishable from the view and the values committed to by \mathcal{A} when receiving a commitment to v using $\langle C, R \rangle$. By the hiding property of $\langle \hat{C}, \hat{R} \rangle$, we then conclude that $\text{nmc}_{\langle C, R \rangle}^{\mathcal{A}}(v, z, \text{id})$ and $\text{nmc}_{\langle \hat{C}, \hat{R} \rangle}^{\mathcal{A}}(v', z, \text{id})$ are indistinguishable.

More formally, R^* internally incorporates a simulated copy of \mathcal{A} and emulates a one-many man-in-the-middle execution by simulating all right receivers and emulating the left $\langle C, R \rangle$ interaction by requesting the appropriate messages expected by \mathcal{A} using $\langle \hat{C}, \hat{R} \rangle$ from outside. For each commitment in the right interactions, if it fails, nothing has to be done. Otherwise, R^* has to internally extract the committed values for each accepting right commitment as long as its tag is different from the left commitment. Depending on whether or not the determining message of this commitment completes before the first message of the left commitment, the extraction is divided into two cases. If yes, there is no need to do extraction, the value (and the corresponding view in this commitment) is given to R^* as auxiliary inputs or non-uniformly hard-wired into R^* .⁷ Otherwise, R^* has to extract the value. This is done as follows: The message scheduling technique guarantees that there exists a point at which the

⁷There is a bit more subtlety here. If the determining message of a right commitment but not the whole commitment completes before the start of the left commitment, then the value committed and the partial joint view of \mathcal{A} in this commitment are also given to R^* as auxiliary inputs. R^* continues the simulation from this partial joint view.

adversary cannot answer the challenge from the receiver simply by “mauling” the commitments on the left as long as it uses a tag different from the left commitment (Otherwise, the value is set to \perp). Thus, R^* keeps rewinding from this point and extracts the value. Through careful analysis, this value is the right one but not the preimage of the one-way function f .⁸

Left is the description of commitment scheme $\langle \hat{C}, \hat{R} \rangle$, which is a variant of $\langle C, R \rangle$. Recall that in the above design of R^* , it needs to extract the value for each right commitment whose determining message is after the start of the left commitment. Thus, the rewinding on the right may rewind CTCom_{sb} in Stage 1 or special-sound \mathcal{WZ} proofs in Stage 3 on the left. We adapt $\langle \hat{C}, \hat{R} \rangle$ to handle the above two cases. Furthermore, our goal is to ensure that from the adversary’s point of view, it is interacting with an honest committer and multiple receivers of $\langle C, R \rangle$, while actually it is interacting with R^* (with the help of \hat{C}). Towards this goal, we make two possible modifications to $\langle C, R \rangle$. First, this one relates to the handling of the rewinding of CTCom_{sb} . Recall that the commitment scheme CTCom_{sb} in Figure 1 is a committed-receiver commitment scheme, where the receiver commits and fixes all of its messages at the start of the protocol. Thus, the rewinding of CTCom_{sb} except the start message has no effect on the properties of $\langle \hat{C}, \hat{R} \rangle$. R^* need interacts with \hat{C} only once in Stage 1 and the response to \mathcal{A} in Stage 1 of left commitment is always the same. Second, this relates to the handling of rewindings of special-sound \mathcal{WZ} proofs. The receiver can ask for an arbitrary number of special-sound \mathcal{WZ} designs in Stage 3. Furthermore, $\langle \hat{C}, \hat{R} \rangle$ does not have a fixed scheduling in Stage 3; the receiver instead gets to choose which design to execute in each iteration (by sending bit i to select design_i). Note that, clearly, any execution of $\langle C, R \rangle$ can be emulated by an execution of $\langle \hat{C}, \hat{R} \rangle$ by simply requesting the appropriate designs. Using a similar analysis as the proof hiding property of $\langle C, R \rangle$, we can show that $\langle \hat{C}, \hat{R} \rangle$ is hiding against expected PPT adversaries.

Concurrent trapdoor. The trapdoor property follows from the concurrent trapdoor property of CTCom_{sb} . We construct a PPT oracle machine $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ such that for any polynomial $p = p(n)$, for any PPT p -concurrent adversary R^* and for all $\mathbb{V} = (v_1, \dots, v_p)$ where $v_1, \dots, v_p \in \{0, 1\}^n$, the two ensembles $\{\text{sta}_{\langle C, R \rangle}^{R^*}(1^n, \mathbb{V})\}_{\mathbb{V}, n, z}$ and $\{(\vec{\tau}, \text{aux}) \leftarrow \text{Ts}_1^{R^*}(1^n, z), \vec{\omega} \leftarrow \text{Ts}_2(\text{aux}, \mathbb{V}) : (\vec{\tau}, \vec{\omega})\}_{\mathbb{V}, n, z}$ are computationally indistinguishable. In the following, for brevity, we use notation $\{\text{Ts}(1^n, \mathbb{V}, z)\}_{\mathbb{V}, n, z}$ to denote the latter ensemble.

More formally, Ts proceeds as follows on input $1^n, \mathbb{V}, z$:

1. Ts_1 interacts with the p -concurrent receiver $R^*(z)$.
2. Ts_1 simulates Stage 1 of CNMCTCom as follows. It invokes the concurrent trapdoor simulator $\widetilde{\text{Ts}} = (\widetilde{\text{Ts}}_1, \widetilde{\text{Ts}}_2)$ for p -concurrent adversary $\hat{R}(z)$ of CTCom_{sb} . Let $(\vec{\tau}, \text{aux}) \leftarrow \widetilde{\text{Ts}}_1(1^n, z')$, where $\vec{\tau} = (\tau_1, \dots, \tau_p)$.
3. Ts_1 simulates Stage 2 by following the honest committer strategy. Denote by $\text{tr}_{i,2}$ the transcript of Stage 2 in the i th commitment.
4. Ts_1 simulates Stage 3 of CNMCTCom as follows. It randomly chooses $v'_1, \dots, v'_p \in \{0, 1\}^n$ and executes $\vec{\omega} \leftarrow \widetilde{\text{Ts}}_2(v'_1, \dots, v'_p, \text{aux})$, where $\vec{\omega} = (\omega_1, \dots, \omega_p)$. For each $i \in [p]$, in the i commitment, Ts_1 runs the special-sound \mathcal{WZ} proofs using ω_i as the witness. Denote by $\text{tr}_{i,3}$ the transcript of Stage 3 in the i th commitment.
5. Update $\tau_i = \tau_i \parallel \text{tr}_{i,2} \parallel \text{tr}_{i,3}$.
6. After all the commit phases compete, Ts_2 is given inputs \mathbb{V} and proceeds as follows. It invokes $\vec{\omega} \leftarrow \widetilde{\text{Ts}}_2(\mathbb{V}, \text{aux})$. For the i th reveal phase, Ts_2 sends ω_i .
7. Output $(\vec{\tau}, \vec{\omega})$.

We show that the output of Ts is indistinguishable from the view of the p -concurrent R^* in real executions. We first design a hybrid argument H . On inputs $1^n, \mathbb{V}, z$, H acts identically as Ts except that in Stage 3, it first invokes $\widetilde{\text{Ts}}_2$ on inputs \mathbb{V} and aux instead of randomly chosen v'_1, \dots, v'_p and aux . Then it uses the returned decommitment information from $\widetilde{\text{Ts}}_2$ as the “fake witness” to complete the \mathcal{WZ} proofs. In the reveal phases, H directly uses the above decommitment information. It follows from the \mathcal{WZ} property in Stage 3 that the outputs of $\text{Ts}(1^n, \mathbb{V}, z)$ and $\text{H}(1^n, \mathbb{V}, z)$ are computationally indistinguishable.

⁸The proof that the extracted value is the right one is different from that in [LPV08]. Suppose, on the contrary, this is not the case. Then we will design an algorithm that breaks the one-wayness of the function f or the \mathcal{WZ} property of the proof in Stage 2.

Next we show that the outputs of $H(1^n, \mathbb{V}, z)$ and $\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ are computationally indistinguishable. Suppose, for contradiction, that this is not the case. Without loss of generality, we assume that R^* is deterministic. Then, there exists a polynomial-time distinguisher D and a polynomial $q(n)$ such that for infinitely many $n \in \mathbb{N}$, there exist strings $v_1, \dots, v_p \in \{0, 1\}^n, z \in \{0, 1\}^*$, such that D distinguishes $\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ and $H(1^n, \mathbb{V}, z)$ with probability at least $\frac{1}{q(n)}$. Fix a generic n for which this happens. We construct a p -concurrent adversary \hat{R} that breaks the concurrent trapdoor property of CTCom_{sb} . \hat{R} on inputs $1^n, z, \mathbb{V}$ proceeds as follows. It internally incorporates R^* and forwards the external commitments in Stage 1 of all commit phases. It follows the honest committer strategy in Stage 2. In Stage 3, \hat{R} receives the decommitment information from external and gives the \mathcal{WI} proofs using these witnesses. In the reveal phases, \hat{R} directly uses the above decommitment information. Finally, \hat{R} outputs whatever R^* outputs. It follows that \hat{R} violates the concurrent trapdoor property of CTCom_{sb} with the same probability that R^* distinguishes between ensemble $\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ and $H(1^n, \mathbb{V}, z)$. \square

B.3.1 The protocol $\text{CNMCTCom}'$

We give a short introduction of the protocol $\text{CNMCTCom}'$ in Remark 5. We claim that $\text{CNMCTCom}'$ is still concurrent non-malleable and concurrent trapdoor. Roughly, it is straightforward that the binding property still holds. The hiding property preserves since the special-sound proofs are still \mathcal{WI} under concurrent composition [FS90]. (we can regard the $4t$ invocations of special-sound \mathcal{WI} proofs as $4t$ concurrent invocations.) For the concurrent trapdoor property, the simulator is almost identical to that of CNMCTCom with the exception that it just ignores the trapdoor information extracted from the CECom_{sh} of Stage 3. In the proof of concurrent non-malleability, the modified commitment scheme $\langle \hat{C}, \hat{R} \rangle$ is almost the same as that of CNMCTCom with the exception that here the receiver is able to query polynomial-many special-sound \mathcal{WI} proofs where the first message is fixed once and for all in each session. Here the receiver is weaker than that of CNMCTCom . Thus, the hiding property of $\langle \hat{C}, \hat{R} \rangle$ still holds.

Another concurrent trapdoor simulator. Considering the CNMZK protocol in Figure 5, we want to use the above concurrent trapdoor simulator in the design of the simulator-extractor for CNMZK . However, the above trapdoor simulator does not suffice upon adaptive corruptions. The main problem is how to handle adaptive corruptions during special-sound \mathcal{WI} proofs. The simulator has to interpret the proof generated using a “fake witness” as one generated using a real witness. Here we solve the above problem by letting the committer generate commitments using concurrent trapdoor simulator of CTCom_{sb} in the special-sound \mathcal{WI} proofs. Accordingly, we define another concurrent trapdoor simulator for $\text{CNMCTCom}'$ which is more suitable to the proof of the CNMZK protocol upon corruptions.

We construct a PPT oracle machine $\text{Ts} = (\text{Ts}_1, \text{Ts}_2)$ such that for any polynomial $p = p(n)$, for any PPT p -concurrent adversary R^* and for all $\mathbb{V} = (v_1, \dots, v_p)$ where $v_1, \dots, v_p \in \{0, 1\}^n$, the two ensembles $\{\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V})\}_{\mathbb{V}, n, z}$ and $\{\text{Ts}(1^n, \mathbb{V}, z)\}_{\mathbb{V}, n, z}$ are computationally indistinguishable.

More formally, Ts proceeds as follows on input $1^n, \mathbb{V}, z$:

1. Ts_1 interacts with the p -concurrent receiver $R^*(z)$.
2. Ts_1 simulates Stage 1 by invoking the concurrent trapdoor simulator $\widetilde{\text{Ts}} = (\widetilde{\text{Ts}}_1, \widetilde{\text{Ts}}_2)$ of CTCom_{sb} . More precisely, it invokes the concurrent extractor \mathcal{E} of CECom_{sh} for p' -concurrent adversary $\hat{R}(z)$, where $p' = p * (1 + 4t)$. Let $\vec{\tau}, \mathbb{V}'$ be the corresponding view and extracted values. Ts_1 uses part of $\vec{\tau}, \mathbb{V}'$ to simulate the Stage 1. Denote by aux_i the auxiliary information in Stage 1 of i th commitment, which includes \mathbb{V}' and the randomness used by Ts_1 .
3. Ts_1 simulates Stage 2 by following the honest committer strategy. Denote by $\text{tr}_{i,2}$ the transcript of Stage 2 in the i th commitment.
4. Roughly, Ts_1 simulates Stage 3 as follows. It uses the concurrent trapdoor simulator of CTCom_{sb} to generate the commitment and answer the query from the adversary. Note that a nice property of Blum’s protocol for Hamiltonicity is that the prover only uses a witness in the final round of the protocol. Thus, we can handle the first two steps a bit easier. More precisely, for simplicity and without loss of generality, we only describe how to simulate Blum’s basic protocol. Denote by G the input.
 - Ts_1 first picks a random permutation π on the vertices and commits to a random graph $G' = \pi(G)$ using the concurrent trapdoor simulator of CTCom_{sb} .

- The adversary picks a random bit challenge b .
- If $b = 0$, Ts_1 sends π along with the revealing of all commitments. Ts_1 invokes the concurrent trapdoor simulator of CTCom_{sb} (with the help of $\vec{\tau}, \mathbb{V}'$) to open the commitments to corresponding values, i.e., for each entry $(i, j) \in G$, it opens the commitment corresponding to $(\pi(i), \pi(j))$ to 1, and for all other entries, it opens to 0. If $b = 1$, Ts_1 chooses a random cycle C and reveals to the adversary only the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C$ (and the adversary checks that all revealed values are 1 and the corresponding entries form a simple n -cycle).

Denote by $\text{tr}_{i,3}$ the transcript of Stage 3 in the i th commitment.

5. Update $\tau_i = \tau_i || \text{tr}_{i,2} || \text{tr}_{i,3}$.
6. After all the commit phases compete, Ts_2 is given inputs \mathbb{V} and proceeds as follows. It invokes $\vec{\omega} \leftarrow \widetilde{\mathsf{Ts}}_2(\mathbb{V}, \text{aux})$. For the i th reveal phase, Ts_2 sends ω_i .
7. Output $(\vec{\tau}, \vec{\omega})$.

We show that the output of Ts is indistinguishable from the view of the p -concurrent R^* in real executions. We first design a hybrid argument H . On inputs $1^n, \mathbb{V}, z$, H acts identically as Ts except that in Stage 1, it runs as honest committer and in the reveal phase, it opens the commitments by revealing the decommitment information in Stage 1. It follows from the concurrent trapdoor property of CTCom_{sb} that the outputs of $\mathsf{Ts}(1^n, \mathbb{V}, z)$ and $H(1^n, \mathbb{V}, z)$ are computationally indistinguishable.

Next we show that the outputs of $H(1^n, \mathbb{V}, z)$ and $\text{sta}_{(C,R)}^{R^*}(1^n, \mathbb{V}, z)$ are computationally indistinguishable. For simplicity, we only consider a basic special-sound \mathcal{WI} proof. Recall that the difference between H and the real executions lies in that the former generates “fake” commitments and equivocates them, whereas the latter generates real commitments and opens them. It follows from the concurrent trapdoor property of CTCom_{sb} that these two outputs are indistinguishable.

B.4 Relations among Commitment Schemes

In the above sections, we give constructions of different kinds of commitment schemes, which will be used in the construction of adaptively secure CNMZK. In order to better understand these commitment schemes, we show the relations among them in Figure 6.

A direct line from A to B and a dashed line from C to B mean that the commitment scheme B is constructed based on the commitment scheme A by replacing part of the primitive in A with the commitment B . For example, the concurrently extractable and trapdoor commitment scheme in Figure 2 is almost the same as the concurrently extractable commitment in [MOSV06] except that the commitment scheme used by the committer is replaced with a concurrent trapdoor one.

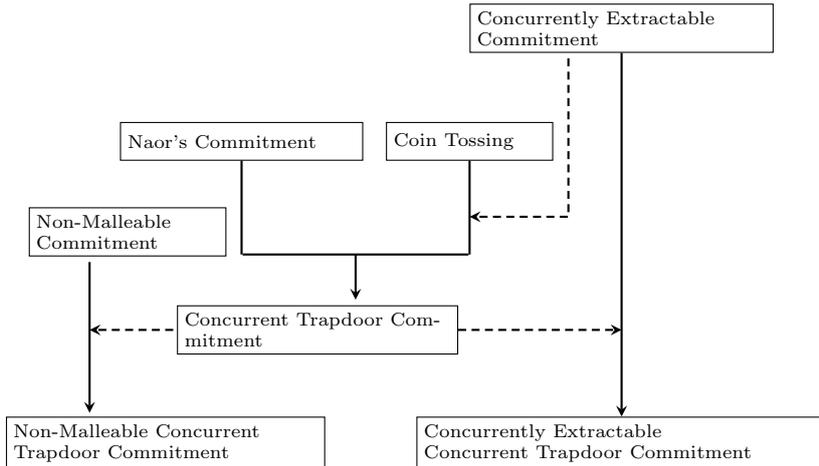


Figure 6: Relations among different kinds of commitment schemes.
 Dashed line denotes the commitment replaced.
 Direct Line denotes the base commitment/tool.

C The Non-Malleability Proof

Proof of Claim 4.1.

Completeness: It is straightforward. An honest prover knows the witness w and commits to w in Stage 2, 3 and 4. Thus, it knows w and the decommitment information of commitments in Stage 2, 3 and 4. Moreover, the Cook-Levin reduction from L' to Hamiltonicity guarantees that given a witness to $x' \in L'$, the prover is able to compute a witness to the corresponding instance in Hamiltonicity. Thus, according to the completeness of MBZKProof, the prover is able to execute the ZK proof and convince the verifier.

Soundness: The soundness property follows from the hiding property of CECom_{sh} , the binding property of CECTCom_{sb} and NMCTCom , and the soundness property of MBZKProof in Stage 6. More formally, if there exists an adversary \mathcal{P}^* such that for infinitely many $n \in \mathbb{N}$, there exist $x \in \{0, 1\}^n \cap \bar{L}$ and a polynomial $p(n)$, such that \mathcal{P}^* convinces an honest verifier the statement x with probability at least $\frac{1}{p(n)}$. We then design an unbounded adversary \mathcal{A} that breaks the soundness property of the protocol MBZKProof for Hamiltonicity. Without loss of generality, we assume that \mathcal{P}^* is deterministic. \mathcal{A} on inputs $x, 1^n$ internally runs a copy of \mathcal{P}^* . \mathcal{A} acts as an honest verifier except that in Stage 6, it forwards the statement x' (i.e., the reduction from $(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)$ to a directed graph G) and messages to an external verifier of MBZKProof. As \mathcal{A} acts as an honest verifier for \mathcal{P}^* in the protocol CNMZKProof, it is straightforward that the probability that \mathcal{A} convinces the external verifier is at least $\frac{1}{p(n)}$. Next we argue that G is not a Hamiltonian graph. We then conclude that \mathcal{A} breaks the soundness property of MBZKProof. Note that the commitment CECom_{sh} in Stage 1 is statistically hiding, \mathcal{P}^* correctly guesses the committed value r of \mathcal{A} after Stage 1 only with negligible probability. Thus, in Stage 2, 3 and 4, \mathcal{P}^* commits to a value r with negligible probability. Moreover, as the CECTCom_{sb} in Stage 2 and the NMCTCom in Stage 3 and 4 are statistically binding, \mathcal{P}^* interprets $\mathcal{T}_2, \mathcal{T}_3$ and \mathcal{T}_4 as commitments to value r only with negligible probability. On the other hand, x is also a false instance for the language L . Thus, $(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)$ is a false instance for the language L' . Due to the property of the Cook-Levin reduction, we conclude that the statement G is not a Hamiltonian graph.

Our simulator \mathcal{S} is similar to the LPTV protocol except the handling of corruptions. On a high level view, \mathcal{S} attempts to simulate the view of \mathcal{A} in one continuously straight-line manner. \mathcal{S} will extract the “fake witnesses” with the help of numerous auxiliary rewinds.⁹ The simulator strategy guarantees that the view of \mathcal{S} depends on the extracted “fake witnesses”, but is otherwise independent of the interaction in auxiliary rewinds.

Note that \mathcal{S} may abort in two manners. At the end of CECom_{sh} , if \mathcal{S} is unable to extract the committed value, \mathcal{S} outputs \perp_{ext} . Or, in Stage 5 of a left interaction, if \mathcal{A} opens its commitment in Stage 1 to a value that is different from the extracted value, \mathcal{S} outputs \perp_{bind} . The following claim bounds the abort probability of \mathcal{S} . The proof is essentially the same as the LPTV protocol [LPTV10].

Claim C.1. \mathcal{S} outputs \perp_{ext} and \perp_{bind} with negligible probability.

C.1 The View Generated by the Simulator

We next show that the view generated by \mathcal{S} is indistinguishable from the real view of \mathcal{A} .

Lemma C.2. *The following ensembles are computationally indistinguishable over $n \in \mathbb{N}$:*

- $\left\{ \mathcal{S} \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0, 1\}^n, id_1, \dots, id_m \in \{0, 1\}^{t(n)}, z \in \{0, 1\}^*}$
- $\left\{ \text{view}_{\mathcal{A}} \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0, 1\}^n, id_1, \dots, id_m \in \{0, 1\}^{t(n)}, z \in \{0, 1\}^*}$

To show Lemma C.2, we introduce a series of hybrid simulators $\{\mathbf{H}_i\}_{0 \leq i \leq m}$. Hybrid \mathbf{H}_i receives the witnesses of the statement proved in all left interactions and proceeds as follows.

⁹The “fake witnesses” here means the extracted value committed by \mathcal{A} in Stage 1. It consists of four parts: r , e_2 for CECTCom_{sb} , \vec{e}_3 and \vec{e}_4 for NMCTCom . In the following, without confusion, we sometimes call r the fake witness.

Real Execution Phase: Run the simulator \mathcal{S} with the man-in-the-middle adaptive adversary \mathcal{A} entirely. Output \perp_{ext} or \perp_{bind} if \mathcal{S} outputs \perp_{ext} or \perp_{bind} . Otherwise, let V be the view of \mathcal{A} produced by \mathcal{S} and $r_j, e_{j,2}, \vec{e}_{j,3}, \vec{e}_{j,4}$ be the values extracted by \mathcal{S} from the Stage 1 of j th left interaction, where r_j is the “fake witness”, and $e_{j,2}, \vec{e}_{j,3}, \vec{e}_{j,4}$ are the corresponding extracted values for the CECTCom_{sb} in Stage 2, NMCTCom in Stage 3 and 4, respectively. (see Remark 1.)

Simulation Phase: Let V_i be the prefix of V up until the i th left interaction has completed Stage 1 of the protocol. Simulate a man-in-the-middle execution with \mathcal{A} , continue from V_i in a “straight-line manner” as follows.

1. Continue of the simulation of right interactions by following the honest verifier strategy (just like \mathcal{S}). Handle adaptive corruptions as \mathcal{S} does.
2. For left interaction $j \leq i$, simulate the interaction in the same manner as \mathcal{S} , i.e., invoke the concurrent trapdoor simulator of CECTCom to generate the commitment in Stage 2, invoke the concurrent trapdoor simulator of NMCTCom to generate the commitment in Stage 3 and 4, and run the proof in Stage 6 using the above decommitment information as the witness. Handle adaptive corruptions as \mathcal{S} does.
3. For left interaction $j > i$, emulate the interaction by following the honest prover strategy. Handle adaptive corruptions by providing \mathcal{A} with the corresponding randomness.

Output Phase: Output \perp_{ext} or \perp_{bind} if \mathcal{S} returns \perp_{ext} or \perp_{bind} . Otherwise, output the newly completed view of \mathcal{A} from the simulation phase.

For $0 \leq i \leq m$, we also define hybrid H_i^+ that proceeds identically as H_i except that in the Simulation Phase, the i th left interaction is simulated by following the honest prover strategy, using the given real witness (rather than using the concurrent trapdoor simulator). Note that these hybrids are only concerned with producing the view of \mathcal{A} , and do not extract the witnesses of the right interactions.

By construction, H_i and H_i^+ abort when \mathcal{S} aborts. Hence by Claim C.1, we get the following claim.

Claim C.3. *For all $0 \leq i \leq m$, H_i and H_i^+ output \perp with negligible probability.*

By Claim C.3, the output of H_0 is statistically close to the real view of \mathcal{A} (they only differ when H_0 aborts, which occurs with negligible probability). The output of H_m , on the other hand, is identical to the output of simulator \mathcal{S} . Therefore, Lemma C.2 follows directly from the Claim C.4 and Claim C.5.

Claim C.4. *For each $i \in [m]$, the following ensembles are statistically indistinguishable over integer $n \in \mathbb{N}$:*

- $\left\{ H_{i-1} \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0,1\}^n, id_1, \dots, id_m \in \{0,1\}^{t(n)}, z \in \{0,1\}^*}$
- $\left\{ H_i^+ \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0,1\}^n, id_1, \dots, id_m \in \{0,1\}^{t(n)}, z \in \{0,1\}^*}$

Proof. Ignoring the fact that H_i^+ and H_{i-1} may abort, their outputs are identical. This is because H_i^+ and H_{i-1} differ only in that when generating the output view, from the end of Stage 1 of $i-1$ st left interaction until the end of Stage 1 of the i th left interactions, H_i^+ employs rewinds. However, these rewinds do not extract any new “fake witnesses” for use in the output view, and do not skew the output distribution because the rewinding schedule is oblivious. Moreover, the corruptions are identically handled in both hybrids. Since both hybrids abort at most with negligible probability by Claim C.3, their outputs are statistically close. □

Claim C.5. *For each $i \in [m]$, the following ensembles are computationally indistinguishable over integer $n \in \mathbb{N}$:*

- $\left\{ H_i^+ \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0,1\}^n, id_1, \dots, id_m \in \{0,1\}^{t(n)}, z \in \{0,1\}^*}$
- $\left\{ H_i \left(1^n, \vec{X}, \vec{ID}, z \right) \right\}_{n \in \mathbb{N}, x_1, \dots, x_m \in \{0,1\}^n, id_1, \dots, id_m \in \{0,1\}^{t(n)}, z \in \{0,1\}^*}$

Proof. The proof of this claim is very different from the LPTV protocol. We has to argue the indistinguishability between the outputs of the two hybrids when adaptive corruptions happen in Stage 2 to Stage 6 of the i th left interaction. The main difference between hybrids H_i^+ and H_i is that the i th left

interaction is generated using a real witness in the former, whereas in the latter it is simulated relying on the concurrent trapdoor simulator of CECTCom_{sb} and NMCTCom . If the adversary \mathcal{A} does not corrupt \mathcal{P}_i or it corrupts other provers or verifiers, then with similar analysis, the outputs of H_i^+ and H_i are computationally indistinguishable by the concurrent trapdoor property of the commitments in Stage 2, 3 and 4, and the strong witness indistinguishability property of the zero-knowledge proof in Stage 6. Otherwise, according to the point where \mathcal{A} corrupts \mathcal{P}_i , we classify several cases. Note that upon corruption, H_i^+ provides \mathcal{A} with the real randomness of \mathcal{P}_i , since it runs as honest prover strategy from Stage 2 to Stage 6.

Corruption in Stage 2: Hybrid H_i provides \mathcal{A} with the randomness for the simulated \mathcal{P}_i using the concurrent trapdoor simulator of CECTCom_{sb} . It follows from the concurrent trapdoor property of CECTCom_{sb} that the outputs of H_i and H_i^+ are computationally indistinguishable.

Corruption in Stage 3: Hybrid H_i invokes the concurrent trapdoor simulator of NMCTCom and CTCom_{sb} to generate the randomness used by \mathcal{P}_i . We again emphasize that H_i need invokes the concurrent trapdoor simulator of NMCTCom and provides the randomness used in the special-sound proof (see the simulator \mathcal{S} for details). It follows from the concurrent trapdoor property of Stage 2 and 3, and the concurrent trapdoor property of CTCom_{sb} in Stage 3 that the outputs of the two hybrids are computationally indistinguishable.

Corruption in Stage 4: H_i handles corruptions as in Stage 3. It follows from the concurrent trapdoor property of Stage 2,3 and 4 that the outputs of the two hybrids are computationally indistinguishable.

Corruption in Stage 6: Recall that H_i uses a fake witness to generate the proof. Upon corruption, it has to explain it as a proof generated using a real witness. It follows from the concurrent trapdoor property of Stage 2, 3 and 4, and the strong \mathcal{WI} property of MBZKProof (along with the adaptive trapdoor property of AIDCom) that the outputs of the two hybrids are computationally indistinguishable.

□

C.2 The Witness Output by the Simulator

Next we show that the extracted witnesses are indeed the NP witnesses of the statements proved in the right interactions. Note that if \mathcal{A} commits to a valid witness using CECTCom_{sb} in Stage 2 of a right interaction, then by Claim C.1, the simulator \mathcal{S} would extract this witness except with negligible probability. We argue the correctness of the output witnesses in the following lemma.

Lemma C.6. *For every probabilistic polynomial-time adaptive adversary \mathcal{A} , for every $n \in \mathbb{N}, x_1, \dots, x_m \in \{0, 1\}^n \cap \mathbb{L}$ and $z \in \{0, 1\}^*$, for every $id_1, \dots, id_m \in \{0, 1\}^{t(n)}$, the probability that \mathcal{A} fails to commit to a valid witness in Stage 2 of a right interaction that is accepting for an honest verifier and uses a different tag from all left interactions, is negligible.*

Our proof follows the structure of the proof of LPTV protocol. The LPTV protocol argues the correctness of the witnesses output by the simulator in the following three steps (Our notation is little different from theirs and we use our notation to describe their result.).

1. In hybrid H_0 (i.e., identical to real execution except with negligible probability), the adversary \mathcal{A} commits to a real witness for each right accepting interaction as long as the verifier is not corrupted and the tag is different from all left interactions.
2. The view and the witnesses output by H_{i-1} and H_i^+ are statistically indistinguishable.
3. The view and the witnesses output by H_i^+ and H_i are computationally indistinguishable.

They then conclude that in H_m (which is the simulator \mathcal{S}), \mathcal{A} commits to a real witness for each right accepting interaction with an uncorrupted verifier that uses a tag different from all left interactions. We also argue the correctness of witnesses output by the simulator following the above three steps. When no corruptions happen, we can alternatively analyze the above three steps in the same manner as the LPTV protocol. But for general cases, we apply a quite different analysis. In the following, we only consider the above three steps when adaptive corruptions happen. The proof for the first two steps are simple. We omit the details here. The tricky part is the third step, which is established in Claim C.7.

Claim C.7. *The view and the witness output by hybrid H_i^+ and H_i are computationally indistinguishable.*

The hybrids H_i and H_i^+ act identically until the completion of Stage 1 of the i th left interaction. According to Lin et al. [LPTV10], this point is called the *cut-off point*. The only difference after the cut-off point between the two hybrids is the remaining i th left interaction, in which H_i uses the “fake witness” (in addition to the concurrent trapdoor simulator of CECTCom_{sb} and NMCTCom), and H_i^+ uses a real witness. For each right interaction, the message scheduling by \mathcal{A} is categorized in the following three cases, see Figure 7.

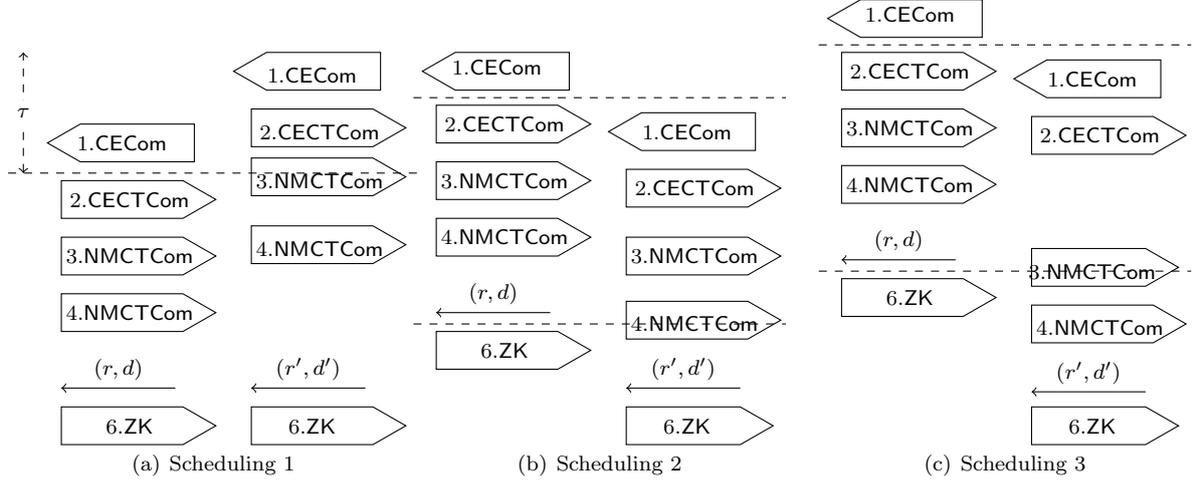


Figure 7: The three scheduling in a man-in-the-middle execution of \mathcal{A}

Scheduling 1: \mathcal{A} completes the Stage 2 commitment on the right before the cut-off point.

Scheduling 2: \mathcal{A} completes the Stage 2 commitment after the cut-off point, but completes the Stage 3 commitment and proof before the start of the Stage 6 of the i th left interaction.

Scheduling 3: \mathcal{A} completes the Stage 2 commitment after the cut-off point, and completes the Stage 3 commitment and proof after the start of the Stage 6 of the i th left interaction.

We define $H_{i,j}$ where $j \in \{1, 2, 3\}$, which acts identically as H_i except that it outputs \perp if the scheduling j does not occur in the output view. Define $H_{i,j}^+$ correspondingly for H_i^+ . The extracted value in each right interaction is defined to be \perp if the interaction is not successfully completed or its tag is the same as one of the left interaction, or the hybrid fails. Define the output of $H_{i,j}^+$ correspondingly for H_i^+ . Next we show that for each right interaction and $j \in \{1, 2, 3\}$, the output of $H_{i,j}$ and $H_{i,j}^+$ are computationally indistinguishable. In the following, we omit the index of the right interaction explicitly, as the following claims hold for every right interaction with an uncorrupted verifier that is accepting and has a tag different from all left interactions.

C.2.1 The Scheduling 1

For the case for Scheduling 1, we get the following claim.

Claim C.8. *For every $0 \leq i \leq m$ the outputs of $H_{i,1}$ and $H_{i,1}^+$ are computationally indistinguishable, i.e.,*

$$\left\{ H_{i,1} \left(1^n, \vec{X}, \text{ID}, z \right) \right\}_{n, \vec{X}, \text{ID}, z} \stackrel{c}{\equiv} \left\{ H_{i,1}^+ \left(1^n, \vec{X}, \text{ID}, z \right) \right\}_{n, \vec{X}, \text{ID}, z}$$

Proof. Since both hybrids act identically before the cut-off point and the commitment in Stage 2 of right interaction is before the cut-off point, the extracted value in Stage 2 of right interaction is identical. By Claim C.5, it follows that the view of the adversary in both hybrids are computationally indistinguishable. Thus, we conclude that the view and the committed value on the right are computationally indistinguishable. \square

C.2.2 The Scheduling 2

For the case for Scheduling 2, we get the following claim.

Claim C.9. *For every $0 \leq i \leq m$, the outputs of $H_{i,2}$ and $H_{i,2}^+$ are computationally indistinguishable, i.e.,*

$$\left\{ H_{i,2} \left(1^n, \vec{X}, \text{ID}, z \right) \right\}_{n, \vec{X}, \text{ID}, z} \stackrel{c}{=} \left\{ H_{i,2}^+ \left(1^n, \vec{X}, \text{ID}, z \right) \right\}_{n, \vec{X}, \text{ID}, z}$$

Proof. Note that Stage 3 to Stage 6 of the right interaction in $H_{i,2}$ and $H_{i,2}^+$ are simulated completely after the cut-off point in a straight-line fashion. It follows from the soundness property of Stage 6 that, except from negligible probability, \mathcal{A} always commits to the same value in Stage 2, 3 and 4 on the right, provided that the right interaction is accepting. Hence, in order to prove Claim C.9, it is sufficient to show that the view and the value v that \mathcal{A} commits to in Stage 3 are indistinguishable. According to the point where \mathcal{A} corrupts \mathcal{P}_i (all other corruptions are handled identically in both hybrids), we distinguish among the following cases: corruption in Stage 2, 3, 4 and 6. We only prove the most complicated case, i.e., corruption in Stage 6. Other cases can be proved in a similar but simpler way. Towards proving this, we introduce a sequence of hybrids $\{H_{i,2}^j\}_{0 \leq j \leq 4}$.

Hybrid $H_{i,2}^0$ proceeds identically to $H_{i,2}^+$ except that in $H_{i,2}^0$, Stage 6 of the i th left interaction is now simulated by invoking the simulator of the ZK protocol MBZKProof. Since the Stage 3 commitment on the right completes before the Stage 6 of the left interaction, the value committed to by the adversary \mathcal{A} in Stage 3 is independent of the ZK proof. Left is the handling of corruptions. Upon corruption, $H_{i,2}^0$ proceeds identically as $H_{i,2}^+$ except that in Stage 6, it has to explain the proof as one generated using real witness.

Recall the simulator of MBZKProof. (The reader is referred to [LZ09] for details.) The ZK simulator first guesses the challenge of the adversary and prepares its message accordingly. If the guess is right, then the simulator proceeds smoothly without difficulty. Otherwise, the simulator keeps on rewinding the adversary and making a new guess until the guess is right. Next we first discuss the simulation in Stage 6 in detail and then describe how $H_{i,2}^0$ handles corruptions in Stage 6. For simplicity, we only give a short description of Blum's basic protocol for Hamiltonicity.

Step 1: $H_{i,2}^0$ makes a random guess $b \in \{0, 1\}$. If $b = 0$, $H_{i,2}^0$ selects a random permutation π of the vertices of G , and commits to the adjacency matrix of the $\pi(G)$ using the algorithm Com of protocol AIDCom. If $b = 1$, $H_{i,2}^0$ creates an adjacency matrix of a graph containing only a random cycle C of length n (all other entries are set to 0). It then uses Com to commit to all 1's in the adjacency matrix. All other entries are filled with commitment values created by the fake algorithm Com'.

Step 2: It receives a challenge bit b' from the adversary \mathcal{A} .

Step 3: If $b \neq b'$, $H_{i,2}^0$ rewinds the adversary to the first step. Otherwise, if $b = b' = 0$, $H_{i,2}^0$ sets the prover's second message to be decommitments to all entries at the adjacency matrix and π ; if $b = b' = 1$, $H_{i,2}^0$ decommits to all 1's in the adjacency matrix (i.e., the cycle $\pi(C)$).

Corruption: Denote by C' the new Hamiltonian cycle in G reduced from the witness w_i (in addition to *real* decommitment information of $\mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ to w_i). If $b = b' = 0$, the first step message has been created exactly as in a real run of the protocol and the simulator can provide \mathcal{A} with the random coins used to create the commitment. If $b = b' = 1$, \mathcal{S} has to make the commitments except the cycle C consistent with the witness C' . More precisely, \mathcal{S} proceeds as follows: find a random permutation π' between the two Hamiltonian cycles C' and C , i.e., $C = \pi'(C')$. Compute $H = \pi'(G)$. For every edge $(u, v) \in C$, it provides \mathcal{A} with the randomness used by Com to commit to 1. For every edge $(u, v) \in H$ and $(u, v) \notin C$, \mathcal{S} uses algorithm Adapt to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 1. For every edge $(u, v) \notin H$, \mathcal{S} uses algorithm Adapt to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 0. \mathcal{S} provides \mathcal{A} with the input w_i and the set of random coins described above as well as π' .

It follows from the adaptive zero-knowledge property of MBZKProof that the view and the value \mathcal{A} commits to is computationally indistinguishable in $H_{i,2}^+$ and $H_{i,2}^0$.

Hybrid $H_{i:2}^1$ proceeds identically to $H_{i:2}^0$ except that in $H_{i:2}^1$, Stage 2 of the i th left interaction is now a commitment generated by the concurrent trapdoor simulator of Ts^2 of $CECTCom_{sb}$. Upon corruption, $H_{i:2}^1$ provides \mathcal{A} with the simulated randomness for \mathcal{P}_i in Stage 2 of the i th left interaction by invoking Ts^2 and opening the commitment in Stage 2 to w_i . Moreover, it uses this decommitment information as part of the witness (in addition to decommitment information in Stage 3 and 4) to handle corruption in the zero-knowledge proof. (all other stages are handled identically as $H_{i:2}^0$.) It follows from the concurrent trapdoor property of $CECTCom_{sb}$ and non-malleability w.r.t $\ell(n)$ -round protocols of $NMCTCom$ (and the fact that Stage 2 of the protocol consist of $\ell(n)$ rounds) that the view and the value \mathcal{A} commits to is computationally indistinguishable in $H_{i:2}^0$ and $H_{i:2}^1$.

Hybrid $H_{i:2}^2$ (and $H_{i:2}^3$ resp.) proceeds identically to $H_{i:2}^1$ (and $H_{i:2}^2$ resp.) except that in $H_{i:2}^2$ (and $H_{i:2}^3$ resp.), Stage 3 (and Stage 4 resp.) of the i th left interaction is now a simulated commitment generated by the concurrent trapdoor simulator Ts^3 (and Ts^4 resp.) of $NMCTCom$. Upon corruption, $H_{i:2}^2$ provides \mathcal{A} with the simulated randomness for \mathcal{P}_i in Stage 3 (and Stage 4 resp.) of the i th left interaction by invoking Ts^3 (and Ts^4 resp.) and opening the commitment in Stage 3 (and Stage 4 resp.) to w_i (in addition to the handling of the special-sound proof). Moreover, in Stage 6, $H_{i:2}^2$ (and $H_{i:2}^3$ resp.) uses this decommitment information as part of the witness to handle corruptions in the zero-knowledge proof. (all other stages are handled identically as $H_{i:2}^1$ (and $H_{i:2}^2$ resp.)) It follows from the concurrent trapdoor property of $NMCTCom$ and the non-malleability w.r.t itself of $NMCTCom$ that the view and the value \mathcal{A} commits to in Stage 3 is computationally indistinguishable in $H_{i:2}^2$ and $H_{i:2}^3$ (and in $H_{i:2}^3$ and $H_{i:2}^4$ resp.).

Hybrid $H_{i:2}^4$ proceeds identically to $H_{i:2}^3$ except that, Stage 6 of the i th left interaction is simulated by proving the Stage 2, 3 and 4 are valid commitments to the value r_i revealed by \mathcal{A} in Stage 5 of this interaction. More precisely, $H_{i:2}^4$ first invokes the concurrent trapdoor simulators of $CECTCom_{sb}$ and $NMCTCom$ and opens the corresponding commitments to r_i . Then $H_{i:2}^4$ uses these decommitment information as the witness to run the zero-knowledge proof in Stage 6 just as the simulator \mathcal{S} does ($H_{i:2}^4$ uses Com' of $AIDCom$ to generate commitments to the adjacency matrix of the permuted graph.). Upon corruption, $H_{i:2}^4$ would explain the commitments in Stage 2, 3 and 4 as to the real witness w_i and explain the proof in Stage 6 as one generated using real witness. Note that, by definition, $H_{i:2}^4$ proceeds identically to the hybrid $H_{i:2}$. Hybrid $H_{i:2}^3$ and $H_{i:2}^4$ differ only in how the proof in Stage 6 of the i th left interaction is simulated and handled upon corruption. In Stage 6, $H_{i:2}^3$ would simulate by invoking the ZK simulator while $H_{i:2}^4$ opens the commitments in Stage 2, 3 and 4 to r_i and uses this as the witness to run the ZK protocol. Upon corruption, both $H_{i:2}^3$ and $H_{i:2}^4$ explain the proof as one generated using real witness w_i . Since in Scheduling 2, the Stage 3 commitment on the right completes before the Stage 6 proof starts, the value \mathcal{A} commits to in Stage 3 is independent of the zero-knowledge proof. Therefore, it follows from the \mathcal{WZ} property of the zero-knowledge protocol and the adaptive trapdoor property of $AIDCom$ that the view and the value \mathcal{A} commits to in Stage 3 are computationally indistinguishable in $H_{i:2}^3$ and $H_{i:2}^4$.

□

Next we formally prove that the outputs of neighbor hybrids above are computationally indistinguishable. Without loss of generality, we only prove the most complicated case where corruptions happen in Stage 6 of the i th left interaction. Other cases can be proved in a similar and simpler way. Note that for the i th left interactions (except Stage 1) of the above hybrids, we only explicitly deal with corruptions of \mathcal{P}_i , since all other corruptions are handled identically in neighbor hybrids.

Claim C.10. *For every $0 \leq i \leq m$, the outputs of $H_{i:2}^+$ and $H_{i:2}^0$ are computationally indistinguishable.*

Proof. Assume, for contradiction, that there exists an adversary \mathcal{A} , a distinguisher D and a polynomial p , such that for infinitely many n , $\vec{X} = (x_1, \dots, x_m)$ where $x_i \in \{0, 1\}^n \cap \mathcal{L}$, $\vec{ID} = (id_1, \dots, id_m)$ where $id_i \in \{0, 1\}^{t(n)}$, $\vec{W} = (w_1, \dots, w_m)$ where $w_i \in \mathcal{R}_{\mathcal{L}}(x_i)$ and $z \in \{0, 1\}^*$, D distinguishes $H_{i:2}^+(1^n, \vec{X}, \vec{ID}, z)$ and $H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)$ with probability at least $\frac{1}{p(n)}$. We then show how this violates the adaptive zero-knowledge property of $MBZKProof$.

Note that the two hybrids $H_{i:2}^+$ and $H_{i:2}^0$ proceed identically before the Stage 6 commitment of the i th left interaction. Therefore, there must exist a partial joint view τ of all parties that defines the execution before Stage 6 of the i th left interaction, such that D distinguishes $\{H_{i:2}^+(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ and

$\{H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ with probability at least $\frac{1}{2p(n)}$, where $\{H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ denotes the outputs of $H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)$ conditioned on the event that the execution is consistent with joint view τ .

We now construct an adversary $\tilde{\mathcal{A}}$ that breaks the adaptive ZK of MBZKProof.¹⁰ $\tilde{\mathcal{A}}$, upon auxiliary inputs $\vec{X}, \vec{W}, \vec{ID}, z, \tau$, internally emulates a man-in-the-middle execution with \mathcal{A} from τ as follows. It emulates the interactions for \mathcal{A} just as $H_{i:2}^+$ with the following exceptions.

- In Stage 6 of the i th left interaction, $\tilde{\mathcal{A}}$ hands the statement $x' = \{(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)\}$ to the external party. It also sends to the external party the “fake” witness to x' . (The non-uniform $\tilde{\mathcal{A}}$ has hard-wired the decommitment information in Stage 2, 3 and 4 or the trapdoor information.) It next relays the message between the external party and \mathcal{A} .
- Upon corruption of \mathcal{P}_i in Stage 6, $\tilde{\mathcal{A}}$ explains the Stage 2, 3 and 4 commitments (in τ) as to w_i and sends the real witness to x' to the external party. Then it forwards the response to \mathcal{A} . All other corruptions are handled identically as $H_{i:2}^+$.

Denote by V the view of $\tilde{\mathcal{A}}$ in the above executions. Denote by v committed to by $\tilde{\mathcal{A}}$. The distinguisher \tilde{D} , on input the view of V and the value v , reconstructs the view $V_{\mathcal{A}}$ and the value v committed to by \mathcal{A} in Stage 3 in emulation by $\tilde{\mathcal{A}}$. Finally, \tilde{D} invokes the distinguisher D on $V_{\mathcal{A}}$ and v , and outputs whatever D outputs. When $\tilde{\mathcal{A}}$ interacts with the prover of MBZKProof, the view $V_{\mathcal{A}}$ and the v are identically distributed to $\{H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)|\tau\}$. When $\tilde{\mathcal{A}}$ interacts with the simulator of MBZKProof, the view $V_{\mathcal{A}}$ and the v are identically distributed to $\{H_{i:2}^+(1^n, \vec{X}, \vec{ID}, z)|\tau\}$. We claim that \tilde{D} distinguishes the two ensembles with probability $\frac{1}{2p(n)}$, which contradicts with the ZK of MBZKProof. \square

Claim C.11. *For every $0 \leq i \leq m$, the outputs of $H_{i:2}^0$ and $H_{i:2}^1$ are computationally indistinguishable.*

Proof. Assume, for contradiction, that there exists an adversary \mathcal{A} , a distinguisher D and a polynomial p , such that for infinitely many n , $\vec{X} = (x_1, \dots, x_m)$ where $x_i \in \{0, 1\}^n \cap L$, $\vec{ID} = (\text{id}_1, \dots, \text{id}_m)$ where $\text{id}_i \in \{0, 1\}^{t(n)}$, $\vec{W} = (w_1, \dots, w_m)$ where $w_i \in R_L(x_i)$ and $z \in \{0, 1\}^*$, D distinguishes $H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)$ and $H_{i:2}^1(1^n, \vec{X}, \vec{ID}, z)$ with probability at least $\frac{1}{p(n)}$. We then show how this violates the robustness of NMCTCom.¹¹

Note that the two hybrids $H_{i:2}^0$ and $H_{i:2}^1$ proceed identically before the Stage 2 commitment of the i th left interaction. It again follows using an averaging argument that, there must exist a partial joint view τ of all parties that defines the execution before Stage 2 of the i th left interaction, such that D distinguishes $\{H_{i:2}^0(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ and $\{H_{i:2}^1(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ with probability at least $\frac{1}{2p(n)}$. Let e_i be the second extracted value (i.e., the trapdoor information of CECTCom_{sb}) in Stage 1 of the i th left interaction. Let ρ be the partial transcript of the CECom_{sh} part in CECTCom_{sb} of the i th left interaction. Consider the machine $\mathcal{B}(0_n)$ and $\mathcal{B}(1_n)$. $\mathcal{B}(0_n)$, on inputs e_i, w_i , continues the execution from ρ and commits to w_i using CECTCom_{sb} . Upon decommitment query, $\mathcal{B}(0_n)$ provides the adversary with the randomness used in generating the commitment. $\mathcal{B}(1_n)$, on inputs e_i, w_i and the partial transcript ρ of the CECom_{sh} part, continues the execution from ρ and generates a commitment using the concurrent trapdoor simulator Ts^2 of CECTCom_{sb} . Upon decommitment query, $\mathcal{B}(1_n)$ opens the commitment to w_i and provides the adversary with the simulated randomness used in generating the commitment with the help of Ts^2 . Due to the concurrent trapdoor property of CECTCom_{sb} , no PPT adversary can distinguish interactions with $\mathcal{B}(0_n)$ and $\mathcal{B}(1_n)$.

We now construct an adversary $\tilde{\mathcal{A}}$ that breaks the non-malleability w.r.t $\mathcal{O}(\ell(n))$ -round protocols of NMCTCom, i.e., the view and the value that $\tilde{\mathcal{A}}$ commits to after interacting with $\mathcal{B}(0_n)$ and $\mathcal{B}(1_n)$ can be distinguished by a distinguisher \tilde{D} . $\tilde{\mathcal{A}}$, upon auxiliary inputs $\vec{X}, \vec{W}, \vec{ID}, z, \tau$, internally emulates a man-in-the-middle execution with \mathcal{A} from τ as follows. It emulates the interactions for \mathcal{A} just as $H_{i:2}^0$ with the following exceptions.

¹⁰We compare between a real execution and a simulated execution. The real execution is slightly different from the execution between an honest prover and an adversary. Here the prover knows the real witness but cheats in the proof.

¹¹The non-malleability requirement of NMCTCom here and the Definition A.7 differ in two aspects. First, in the commitment on the left interaction, the former is either a regular commitment or a simulated commitment generated by the trapdoor simulator, moreover, the trapdoor information (as auxiliary input) is known in advance, whereas the latter is a regular commitment to either values. Second, the internal randomness of the committer on the left might be revealed upon corruption in the former (both commitments opens to the same value), whereas it is hidden in the latter. Due to the trapdoor property of NMCTCom, we claim that the non-malleability property still preserves in this case and the proof in Appendix B.3 can be modified to adapt this setting.

- To emulate the Stage 2 of the i th left interaction, it externally sends \mathcal{B} the witness w_i and the extracted value e_i . It next forwards the commitment from \mathcal{B} to \mathcal{A} .
- In Stage 3 of the right interaction, it externally forwards messages from \mathcal{A} to an honest receiver of NMCTCom.
- Upon corruption of \mathcal{P}_i in Stage 2 and afterwards, $\tilde{\mathcal{A}}$ sends the decommitment query to \mathcal{B} and forwards the response to \mathcal{A} . All other corruptions are handled identically as $H_{i:2}^0$.

Denote by \mathbb{V} the view of $\tilde{\mathcal{A}}$ in the above executions. Denote by v committed to by $\tilde{\mathcal{A}}$. The distinguisher \tilde{D} , on input the view of \mathbb{V} and the value v , reconstructs the view $\mathbb{V}_{\mathcal{A}}$ and the value v committed to by \mathcal{A} in Stage 3 in emulation by $\tilde{\mathcal{A}}$. Finally, \tilde{D} invokes the distinguisher D on $\mathbb{V}_{\mathcal{A}}$ and v , and outputs whatever D outputs. When $\tilde{\mathcal{A}}$ interacts with $\mathcal{B}(b_n)$, the view $\mathbb{V}_{\mathcal{A}}$ and the v are identically distributed to $\{H_{i:2}^b(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$. We claim that \tilde{D} distinguishes the view and the values committed by $\tilde{\mathcal{A}}$ using NMCTCom with probability $\frac{1}{2p(n)}$, which contradicts with the robustness of NMCTCom. \square

Claim C.12. *For every $0 \leq i \leq m$, the outputs of $H_{i:2}^1$ and $H_{i:2}^2$ are computationally indistinguishable.*

Proof. Assume, for contradiction, that there exists an adversary \mathcal{A} , a distinguisher D and a polynomial p , such that for infinitely many n , $\vec{X} = (x_1, \dots, x_m)$ where $x_i \in \{0, 1\}^n \cap \mathbb{L}$, $\vec{W} = (w_1, \dots, w_m)$ where $w_i \in \mathbb{R}_{\mathbb{L}}(x_i)$, $\vec{\text{ID}} = (\text{id}_1, \dots, \text{id}_m)$ where $\text{id}_i \in \{0, 1\}^{t(n)}$, and $z \in \{0, 1\}^*$, D distinguishes $H_{i:2}^1(1^n, \vec{X}, \vec{\text{ID}}, z)$ and $H_{i:2}^2(1^n, \vec{X}, \vec{\text{ID}}, z)$ with probability at least $\frac{1}{p(n)}$. We then show how this violates the non-malleability w.r.t itself of NMCTCom.

Note that the two hybrids $H_{i:2}^1$ and $H_{i:2}^2$ proceed identically before the Stage 3 commitment of the i th left interaction is sent. Using an averaging argument, it follows that there must exist a partial joint view τ of all parties that defines the execution before Stage 3 of the i th left interaction, such that D distinguishes $\{H_{i:2}^1(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$ and $\{H_{i:2}^2(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$ with probability at least $\frac{1}{2p(n)}$. Let \vec{e}_i be the third part of the extracted values (i.e., the trapdoor information of NMCTCom) in Stage 1 of the i th left interaction.

We now construct an adversary $\tilde{\mathcal{A}}$ that breaks the non-malleability w.r.t itself of NMCTCom. $\tilde{\mathcal{A}}$, upon auxiliary inputs $\vec{X}, \vec{W}, \vec{\text{ID}}, z, \tau$, internally emulates a man-in-the-middle execution with \mathcal{A} from τ as follows. It emulates the interactions for \mathcal{A} just as $H_{i:2}^1$ with the following exceptions.

- To emulate the Stage 3 of the i th left interaction, it externally sends the committer (i.e., honest committer or trapdoor simulator) of NMCTCom_{sb} the witness w_i and the extracted value \vec{e}_i . It next forwards the commitment from the external committer to \mathcal{A} .
- In Stage 3 of the right interaction, it externally forwards messages from \mathcal{A} to an honest receiver of NMCTCom.
- Upon corruption of \mathcal{P}_i in Stage 3 and afterwards, $\tilde{\mathcal{A}}$ asks the external committer of NMCTCom to provide \mathcal{A} with the internal randomness, i.e., decommitment information. All other corruptions are handled identically as $H_{i:2}^1$.

Denote by \mathbb{V} the view of $\tilde{\mathcal{A}}$ in the above executions. Denote by v committed to by $\tilde{\mathcal{A}}$. The distinguisher \tilde{D} , on input the view of \mathbb{V} and the value v , reconstructs the view $\mathbb{V}_{\mathcal{A}}$ and value v committed to by \mathcal{A} in the emulation by $\tilde{\mathcal{A}}$. Finally, \tilde{D} invokes the distinguisher D on $\mathbb{V}_{\mathcal{A}}$ and the committed value v , and outputs whatever D outputs. From the construction, when $\tilde{\mathcal{A}}$ interacts with real committer of NMCTCom, the view $\mathbb{V}_{\mathcal{A}}$ and v is identically distributed to $\{H_{i:2}^1(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$. When $\tilde{\mathcal{A}}$ interacts with trapdoor simulator (knowing trapdoor in advance) of NMCTCom, the view $\mathbb{V}_{\mathcal{A}}$ and v is identically distributed to $\{H_{i:2}^2(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$. We claim that \tilde{D} distinguishes the view and the value committed by $\tilde{\mathcal{A}}$ using NMCTCom with probability $\frac{1}{2p(n)}$, which contradicts with the non-malleability w.r.t itself of NMCTCom. \square

It follows from the same argument as in $H_{i:2}^2$ that the following claim holds.

Claim C.13. *For every $0 \leq i \leq m$, the outputs of $H_{i:2}^2$ and $H_{i:2}^3$ are computationally indistinguishable.*

Claim C.14. *For every $0 \leq i \leq m$, the outputs of $H_{i:2}^3$ and $H_{i:2}^4$ are computationally indistinguishable.*

Proof. Assume, for contradiction, that there exists an adversary \mathcal{A} , a distinguisher D and a polynomial p , such that for infinitely many n , $\vec{X} = (x_1, \dots, x_m)$ where $x_i \in \{0, 1\}^n \cap \mathbb{L}$, $\vec{W} = (w_1, \dots, w_m)$ where

$w_i \in R_L(x_i)$, $\vec{\text{ID}} = (\text{id}_1, \dots, \text{id}_m)$ where $\text{id}_i \in \{0, 1\}^{t(n)}$ and $z \in \{0, 1\}^*$, D distinguishes $H_{i,2}^3(1^n, \vec{X}, \vec{\text{ID}}, z)$ and $H_{i,2}^4(1^n, \vec{X}, \vec{\text{ID}}, z)$ with probability at least $\frac{1}{p(n)}$. We then show how this violates the adaptive zero-knowledge property of MBZKProof.¹²

Note that the two hybrids $H_{i,2}^3$ and $H_{i,2}^4$ proceed identically before the Stage 6 proof of the i th left interaction is sent. Using an averaging argument, it follows that there must exist a partial joint view τ of all parties that defines the execution before Stage 6 of the i th left interaction, such that D distinguishes $\{H_{i,2}^3(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$ and $\{H_{i,2}^4(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$ with probability at least $\frac{1}{2p(n)}$. Let dec_{real} (dec_{fake} resp.) be the whole decommitments information to w_i (to r_i opened by the adversary in Stage 5 resp.) in Stage 2, 3 and 4 of the i th left interaction. Let v be the value committed to by \mathcal{A} on the right.

We now construct an adversary $\tilde{\mathcal{A}}$ such that the view of the adversary after receiving a simulated MBZKProof proof (as in $H_{i,2}^3$) and a “real” proof (as in $H_{i,2}^4$) can be distinguished by a distinguisher \tilde{D} . $\tilde{\mathcal{A}}$, upon auxiliary inputs $\vec{X}, \vec{W}, \vec{\text{ID}}, z, \tau$, internally emulates a man-in-the-middle execution with \mathcal{A} from τ as follows. $\tilde{\mathcal{A}}$ proceeds as $H_{i,2}^3$ with the following exceptions.

- To emulate the Stage 6 of the i th left interaction, it externally sends the party the witnesses to $x' = \{(x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)\}$, i.e., the witness reduced from dec_{fake} . It next forwards the proof from the external to \mathcal{A} .
- Upon corruption of \mathcal{P}_i in Stage 6, $\tilde{\mathcal{A}}$ explains the Stage 2, 3 and 4 commitments (in τ) to w_i and sends the real witness to x' to the external party, i.e., the witness reduced from dec_{real} . $\tilde{\mathcal{A}}$ asks the external party to provide the internal \mathcal{A} with the randomness, i.e., randomness used in generating the permuted graph in MBZKProof. All other corruptions are handled identically as $H_{i,2}^3$.

Denote by \mathbf{V} the joint view of all parties in the above executions. The distinguisher \tilde{D} , on input the view of \mathbf{V} , and the value v committed to by $\tilde{\mathcal{A}}$, reconstructs the view $\mathbf{V}_{\mathcal{A}}$ and the value v committed to by \mathcal{A} in the emulation of $\tilde{\mathcal{A}}$. \tilde{D} then invokes the distinguisher D on $\mathbf{V}_{\mathcal{A}}$ and the committed value v , and outputs whatever D outputs. From the construction, when $\tilde{\mathcal{A}}$ interacts with the external simulator, the view $\mathbf{V}_{\mathcal{A}}$ and the v is identically distributed to $\{H_{i,2}^3(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$. When $\tilde{\mathcal{A}}$ interacts with the external prover, the view $\mathbf{V}_{\mathcal{A}}$ and the v is identically distributed to $\{H_{i,2}^4(1^n, \vec{X}, \vec{\text{ID}}, z)|\tau\}$. It follows that \tilde{D} distinguishes the view of $\tilde{\mathcal{A}}$ using MBZKProof with probability $\frac{1}{2p(n)}$, which contradicts the adaptive zero-knowledge property of MBZKProof (actually, the adaptive trapdoor property of AIDCom). \square

C.2.3 The Scheduling 3

For the case for Scheduling 3, we get the following claim.

Claim C.15. *For every $0 \leq i \leq m$, the outputs of $H_{i,3}$ and $H_{i,3}^+$ are computationally indistinguishable, i.e.,*

$$\left\{ H_{i,3} \left(1^n, \vec{X}, \vec{\text{ID}}, z \right) \right\}_{n, \vec{X}, \vec{\text{ID}}, z} \stackrel{c}{\equiv} \left\{ H_{i,3}^+ \left(1^n, \vec{X}, \vec{\text{ID}}, z \right) \right\}_{n, \vec{X}, \vec{\text{ID}}, z}$$

Proof. Due to the soundness property of zero-knowledge protocol, the man-in-the-middle adversary always commits to the same value in Stage 2, 3 and 4. As the Stage 3 of right interaction completes after the Stage 6 of the i th left interaction, the Stage 4 of right interaction starts after the Stage 6 of the i th left interaction. It follows from the non-malleability w.r.t $\omega(1)$ -round protocols of NMCTCom and the strong \mathcal{WI} property of the zero-knowledge protocol (in addition to the concurrent trapdoor property of CECTCom_{sb} and NMCTCom) that the outputs of $H_{i,3}$ and $H_{i,3}^+$ are computationally indistinguishable.

More formally, according to the point where \mathcal{A} corrupts \mathcal{P}_i , we distinguish among the following cases: corruption in Stage 2, 3, 4 and 6. We only prove the most complicated case, i.e., corruption in Stage 6. Other cases can be proved in a similar but simpler way. Note that we only explicitly deal with corruption of \mathcal{P}_i in the i th left interaction (except Stage 1), since all other corruptions are handled identically in hybrids $H_{i,3}$ and $H_{i,3}^+$.

Assume, for contradiction, that there exists an adversary \mathcal{A} , a distinguisher D and a polynomial p , such that for infinitely many n , $\vec{X} = (x_1, \dots, x_m)$ where $x_i \in \{0, 1\}^n \cap L$, $\vec{W} = (w_1, \dots, w_m)$ where

¹²Here the MBZKProof is ZK between a “real” execution and a simulated execution. In the real execution, the prover generates fake commitments to a permuted graph using Com' of AIDCom and answers the challenges of the adversary using Adapt of AIDCom with witness w_1 . Upon corruptions, the prover provides the adversary with the randomness which explains the messages as generated using another witness w_2 (see the simulation strategy of S).

$w_i \in R_L(x_i)$, $\vec{ID} = (\text{id}_1, \dots, \text{id}_m)$ where $\text{id}_i \in \{0, 1\}^{t(n)}$, and $z \in \{0, 1\}^*$, D distinguishes $H_{i,3}(1^n, \vec{X}, \vec{ID}, z)$ and $H_{i,3}^+(1^n, \vec{X}, \vec{ID}, z)$ with probability $\frac{1}{p(n)}$. We then show how this violates the non-malleability w.r.t $\omega(1)$ -round protocols of NMCTCom.

Towards this goal, first note that the two experiment $H_{i,3}$ and $H_{i,3}^+$ proceed identically before the Stage 2 commitment of the i th left interaction. Using an averaging argument, there must exist a partial joint view τ of all parties that defines the execution before Stage 2 of the i th left interaction, such that D distinguishes $\{H_{i,3}(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ and $\{H_{i,3}^+(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ with probability at least $\frac{1}{2p(n)}$. Let $r_i, e_{i,2}, \vec{e}_{i,3}, \vec{e}_{i,4}$ be the extracted values in Stage 1 of the i th left interaction. Consider the machine $\mathcal{B}(0_n)$ which upon receiving w_i , provides a commitment to CECTCom_{sb} and two commitments to NMCTCom to w_i honestly (Recall that part of these commitments are in Stage 1.). Upon decommitment query, $\mathcal{B}(0_n)$ provides the internal randomness of committer in these commitments. Consider another machine $\mathcal{B}(1_n)$ which upon receiving $r_i, e_{i,2}, \vec{e}_{i,3}, \vec{e}_{i,4}$, provides a commitment CECTCom_{sb} and two commitments to NMCTCom using all the concurrent trapdoor simulator accordingly. Upon decommitment query, $\mathcal{B}(1_n)$ explains these commitments as to w_i . It follows from the concurrent trapdoor property of CECTCom_{sb} and NMCTCom that no PPT adversary can distinguish between $\mathcal{B}(0_n)$ and $\mathcal{B}(1_n)$.

We now construct an adversary $\tilde{\mathcal{A}}$. $\tilde{\mathcal{A}}$, upon receiving $w_i, r_i, e_{i,2}, \vec{e}_{i,3}, \vec{e}_{i,4}$ and τ as auxiliary inputs, internally emulates a man-in-the-middle execution with \mathcal{A} from τ as follows. $\tilde{\mathcal{A}}$ proceeds as $H_{i,3}$ with the following exceptions.

- To emulate the Stage 2, 3 and 4 of the i th left interaction, it externally sends \mathcal{B} the values $w_i, r_i, e_{i,2}, \vec{e}_{i,3}, \vec{e}_{i,4}$. It next forwards the messages from \mathcal{B} to \mathcal{A} .
- Upon corruptions of \mathcal{P}_i in Stage 2, 3 or 4, $\tilde{\mathcal{A}}$ sends decommitment query to \mathcal{B} and forwards the messages from \mathcal{B} to $\tilde{\mathcal{A}}$, i.e., the simulated randomness of \mathcal{P}_i . All other corruptions are handled identically as $H_{i,3}$.
- $\tilde{\mathcal{A}}$ aborts interactions with \mathcal{A} at the outset of the Stage 6 of the i th left interaction.

Denote by V_{part} the joint view of all parties in the above executions. Denote by $x' = (x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)$ the proof statement in Stage 6 of the i th left interaction. Denote by w' the witness for the statement $x' \in L'$ in the above execution. If $\tilde{\mathcal{A}}$ interacts with $\mathcal{B}(0_n)$, then consider the machine $B(0_n)$, which upon receiving x', w' and the trapdoors $e_2, \vec{e}_3, \vec{e}_4$ as auxiliary inputs, generates a MBZKProof proof honestly. If $\tilde{\mathcal{A}}$ interacts with $\mathcal{B}(1_n)$, then consider the machine $B(1_n)$, which upon receiving x', w' and the trapdoors $r, e_2, \vec{e}_3, \vec{e}_4$ as auxiliary inputs, generates a MBZKProof proof as \mathcal{S} . Upon corruptions, $B(1_n)$ uses the trapdoors to explain the commitment in Stage 2, 3 and 4 as to w_i and explain the MBZKProof proof as one generated using the real witness. It follows from the strong \mathcal{WI} property of MBZKProof that no PPT distinguisher can distinguish interactions with $B(0_n)$ and $B(1_n)$.

Next we design an adversary \mathcal{A}' such that the view and the value that \mathcal{A}' commits to after interacting with $B(0_n)$ and $B(1_n)$ can be distinguished by a distinguisher \tilde{D} . \mathcal{A}' on input the view of V_{part} , first emulates the interactions for \mathcal{A} from the view V_{part} . Then \mathcal{A}' continues the executions of the remaining interactions for \mathcal{A} . It emulates the left provers and right verifiers for \mathcal{A} just as $H_{i,3}$ with the following exceptions.

- To emulate the Stage 6 of the i th left interaction, \mathcal{A}' sends the proof statement $x' = (x, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, r)$ to the external B . Then \mathcal{A}' forwards the proof to \mathcal{A} . Here we emphasize that B is given the private information of prover in the view V_{part} , i.e., $w', e_2, \vec{e}_3, \vec{e}_4$.
- In Stage 4 of the right interaction, it externally forwards messages from \mathcal{A} to an honest receiver of NMCTCom.
- Upon corruption of \mathcal{P}_i in Stage 6 of the i th left interaction, \mathcal{A}' asks B to provide the simulated randomness of \mathcal{P}_i . All other corruptions are handled identically as $H_{i,3}$.

The distinguisher \tilde{D} , on input the view and the values v committed to by \mathcal{A}' , reconstructs the view $V_{\mathcal{A}}$ and value v committed to by \mathcal{A} in Stage 4 of the above executions. \tilde{D} then invokes the distinguisher D on the constructed view and the value and outputs whatever D outputs. Note that if $\tilde{\mathcal{A}}$ interacts with $\mathcal{B}(0_n)$ ($\mathcal{B}(1_n)$ resp.), then the reconstructed view $V_{\mathcal{A}}$ and the committed value v are identically distributed to $\{H_{i,3}^+(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ ($\{H_{i,3}(1^n, \vec{X}, \vec{ID}, z)|\tau\}$ resp.). It follows that \tilde{D} distinguishes the view and the value committed to by \mathcal{A}' using NMCTCom with probability $\frac{1}{2p(n)}$, which contradicts the non-malleability w.r.t $\omega(1)$ -round protocols of NMCTCom. □

D Blum's Basic Protocol for Hamiltonicity

Following is the description of Blum's basic protocol for Hamiltonicity [Blu86]. The protocol is a zero-knowledge proof with soundness error $\frac{1}{2}$. By parallel execution of the basic protocol, the soundness error can be reduced to 2^{-n} . However, the zero-knowledge property is lost.

Common Input: A directed graph $G = (V, E)$ with $n \stackrel{\text{def}}{=} |V|$.

Auxiliary Input to Prover: A directed Hamiltonian cycle, $C \subset G$, in G

Step 1: The prover selects a random permutation π of the vertices V , and commits (using a perfectly-binding commitment scheme) the entries of the adjacency matrix of the resulting permuted graph. That is, it sends an n -by- n matrix of commitments so that the $(\pi(i), \pi(j))^{\text{th}}$ entry is a commitment to 1 if $(i, j) \in E$, and is a commitment to 0 otherwise.

Step 2: The verifier sends a random chosen bit $b \in \{0, 1\}$.

Step 3: If $b = 0$ then the prover sends π to the verifier along with the revealing of all commitments (and the verifier checks that the revealed graph is indeed isomorphic to G via π); If $b = 1$, the prover reveals to the verifier only the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C$ (and the verifier checks that all revealed values are 1 and the corresponding entries form a simple n -cycle). In both cases check that the decommitments are valid. Accept if only if the corresponding conditions holds.

Figure 8: Blum's basic protocol for Hamiltonicity