Adaptive Concurrent Non-Malleability with Bare Public-Keys*

Andrew C. Yao† Moti Yung‡ Yunlei Zhao§

Abstract

Concurrent non-malleability (CNM) is central for cryptographic protocols running concurrently in environments such as the Internet. In this work, we formulate CNM in the bare public-key (BPK) model, and show that round-efficient concurrent non-malleable cryptography with *full* adaptive input selection can be established, in general, with bare public-keys (where, in particular, no trusted assumption is made).

1 Introduction

Concurrent non-malleability is central for cryptographic protocols secure against concurrent man-in-the-middle (CMIM) attacks. In the CMIM setting, polynomially many concurrent executing instances (sessions) of a protocol take place in an asynchronous setting (appropriate for environments such as over the Internet), and all the unauthenticated communication channels (among all the concurrent sessions) are controlled by a probabilistic polynomial-time (PPT) CMIM adversary $\mathcal{A}$. In this setting, honest players are assumed oblivious of each other's existence, nor do they generally know the topology of the network, and thus cannot coordinate their executions. The CMIM adversary $\mathcal{A}$ (controlling the communication channels) can do whatever it wishes. When CNM with adaptive input selection is considered, $\mathcal{A}$ can also set input to each session.

Unfortunately, in the stringent CMIM setting, large classes of cryptographic functionalities cannot be securely implemented round-efficiently, and even cannot be securely implemented with *non-constant* round-complexity against *adaptive input selecting* CMIM adversaries in the plain model [13, 56, 54]. In such cases, some setup assumptions are necessary, and establishing the general feasibility of round-efficient concurrent non-malleable cryptography with adaptive input selection, with setups as minimal as possible, has been being a basic problem extracting intensive research efforts in the literature.

In this work, we investigate CNM security in the bare public-key model (introduced by Canetti, Goldreich, Goldwasser and Micali [12]). A protocol in the BPK model simply assumes that all players have each deposited a public-key in a public file before any interaction takes place among the users. Note that, no assumption is made on whether the public-keys deposited are unique or valid (i.e., public keys can even be "nonsensical," where no corresponding secret-keys exist or are known) [12]. That is, no trusted third party is assumed, the underlying communication network is assumed to be adversarially asynchronous, and preprocessing is reduced to minimally non-interactively posting public-keys in a public file. In many cryptographic settings, availability of a public key infrastructure (PKI) is assumed or required and in these settings the BPK model is, both, natural and attractive (note that the BPK model is, in fact, a weaker version of PKI where in the later added key certification is assumed). It was pointed out by Micali and Reyzin [59] that BPK is, in fact, applicable to interactive systems in general.

*The work described in this paper was supported in part by the National Basic Research Program of China Grant (973) 2007CB807900, 2007CB807901, and by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Number CityU 122105) and CityU Research Grant (9380039). The third author is also supported by a grant from the National Natural Science Foundation of China NSFC (No. 60703091).

†Institute for Theoretical Computer Science (ITCS), Tsinghua University, Beijing, China. andrewcyao@tsinghua.eud.cn

‡Google Inc. and Columbia University, New York, NY, USA. moti@cs.columbia.edu

§Software School, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn Works partially done while visiting Tsinghua university and City University of Hong Kong.

1.1 Our contributions

We examine concurrent non-malleability in the BPK model, by investigating two types of protocols, specifically, zero-knowledge (ZK) [44] and coin-tossing (CT) [9], both of which are central and fundamental to modern cryptography.

We show the insufficiency of existing CNM formulations in the public-key (not necessarily bare public-key) model, reformulate CNM zero-knowledge (CNMZK) and CNM coin-tossing (CNMCT) in the BPK model. The CNMCT definition implies, or *serves as a general basis to formulate*, the CNM security for any cryptographic protocol in the BPK model against CMIM with full adaptive input selection. By CMIM with *full* adaptive input selection, we mean that the CMIM adversary can set inputs to all concurrent sessions; furthermore and different from the traditional formulation of adaptive input selection, the adversary does not necessarily set the input to each session at the beginning of the session; Rather, the input may be set on the way of the session, and is based on the whole transcript evolution (among other concurrent sessions and the current session). We motivate the desirability of achieving CNM security against CMIM with full adaptive input selection, and make in-depth discussions.

We then present a constant-round CNMCT protocol in the BPK model under standard assumptions, which is enabled by the Pass-Rosen ZK (PRZK) result [67, 68]. The importance of the CNMCT protocol is that it can be used to transform concurrent non-malleable protocols that are originally developed in the common random string (CRS) model into the weaker BPK model (with full adaptive input selection). That is, *round-efficient* concurrent non-malleable cryptography *with full adaptive input selection* can be established with bare public-keys, *in general*.

For space limitation, discussions on related works are given in Appendix A.

2 Preliminaries

We briefly recall the preliminaries, with details given in Appendix B and C.

The CMIM setting in the BPK model. We briefly describe the CMIM setting in the BPK model for any two-party protocol $\langle L, R\rangle$ with players of fixed roles, where L stands for the left-player (e.g., the prover) and R stands for the right-player (e.g., the verifier). The formalization can be directly extended to the general case of interchangeable roles. For more details, the reader is referred to Appendix B.

Each player in the BPK model works in two stages: the *key-generation stage* in which it generates and registers a public-key in a public file F; and the *proof* stage between two players specified by a key pair (PK_L, PK_R) in F. The file F output at the end of the key-generation stage is denoted as: $\{PK_I^{(1)}, PK_I^{(2)}, \cdots, PK_I^{(poly(n))}\}$ that is to be used and *remain intact* during the proof stage, where $PK_I^{(j)}$ denotes a left-player key if $I = L$ or a right-player key if $I = R$. We also denote by $\mathcal{R}^I_{KEY}$ the $\mathcal{NP}$-relation validating the key pair (PK_I, SK_I), i.e., whether SK_I is a valid secret-key corresponding to PK_I.

The CMIM adversary $\mathcal{A}$. In the key-generation stage, on 1^n and some auxiliary input $z \in \{0,1\}^*$ and a pair of honestly generated public-keys (PK_L, PK_R), $\mathcal{A}$ outputs a set of public-keys F'. Then the public file F for the proof stage is set to be $F' \cup \{PK_L, PK_R\}$.

In the proof stage, $\mathcal{A}$ can concurrently interact with any polynomial number of instances of the honest left-player of public-key PK_L in the *left CMIM interaction part.* The interactions with each instance of the honest left-player is called a *left session*, in which $\mathcal{A}$ plays the role of the right-player with a public-key $PK_R^{(j)} \in F$; *Simultaneously*, $\mathcal{A}$ interacts with any polynomial number of instances of the honest right-player of public-key PK_R in the *right CMIM interaction part.* The interactions with each instance of the honest right-player is called a *right session*, in which $\mathcal{A}$ plays the role of the left-player with a public-key $PK_L^{(j)} \in F$. For CMIM-adversary with adaptive input selection, $\mathcal{A}$ can further set the inputs to left sessions adaptively based on its view (besides adaptively setting inputs to right sessions).

In all cases, the honest player instances answer messages from $\mathcal{A}$ promptly, and use independent random-tapes in different sessions of the proof stage. The adversary's goal is to complete a right session such that it would not have done so without involving the CMIM interactions. A CMIM adversary is called s-CMIM adversary, for a positive polynomial $s(\cdot)$, if the adversary involves, on security parameter 1^n, at most $s(n)$ concurrent sessions in each CMIM interaction part and registers at most $s(n)$ public-keys in F'.

For any $(PK_L, SK_L) \in \mathcal{R}^L_{KEY}$ and $(PK_R, SK_R) \in \mathcal{R}^R_{KEY}$, we denote by $view_{\mathcal{A}}^{L(SK_L),R(SK_R)}(1^n, z, PK_L, PK_R)$ the random variable describing the view of $\mathcal{A}$ specific to (PK_L, PK_R), which includes its random tape, the auxiliary string z, the (specific) (PK_L, PK_R), and all messages it receives from the instances of $L(1^n, SK_L)$ and $R(1^n, SK_R)$ in the proof stages.

Building tools. *Pseudorandom functions* (PRF) can be constructed under any one-way function (OWF) [40, 38]. A OWF $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is called *linear*, if for sufficiently large n and any $x \in \{0,1\}^n$ $|f(x)| = O(n)$, where $|f(x)|$ denotes the length of $f(x)$.

Non-interactive statistically-binding commitments can be based on any one-way permutation (OWP) [9, 42]. Practical perfectly-binding non-interactive (string) commitment scheme can be based on the decisional Diffie-Hellman (DDH) assumptions [31]. A commitment scheme C is called *linear*, if for sufficiently large n and any string $x \in \{0,1\}^n$ both $|C(x,s)|$ (i.e., the length of the commitment to x using random coins s) and $|s|$ (i.e., the length of s) are bounded by $O(n)$. In particular, the perfectly-binding commitment scheme from [31] is *linear*.

Very roughly, *adaptive tag-based one-left-many-right non-malleable statistical zero-knowledge argument of knowledge* (NMSZKAOK) is non-malleable against any one-left-many-right PPT man-in-the-middle adversary $\mathcal{A}$ who involves one left session with the prover and many right sessions with verifiers; each session is indexed by a string (called a tag) and $\mathcal{A}$ is allowed to set the input and tag to the left session (besides those of right sessions). Then, the security says that for any PPT one-left-many-right MIM adversary $\mathcal{A}$, there exists an (expected) polynomial-time simulator S such that S output a simulated transcript that is *statistically* indistinguishable from the real view of $\mathcal{A}$; Moreover, for any successful right session on a input in the simulated transcript w.r.t. a tag different from that of the left session, a valid $\mathcal{NP}$-witness (to the statement of this session selected adaptively by $\mathcal{A}$) is also extracted.

We note that the Pass-Rosen ZK (PRZK, in short) [67, 68], *with some specified length parameters* $l(n)$ where $l(\cdot)$ is a positive polynomial and n is the security parameter, is the only known solution for *constant-round* adaptive tag-based one-left-many-right NMSZKAOK. Furthermore, PRZK is *public-coin* and can be *perfect* ZK. In [67, 68], the tag and input length is just specified to be the security parameter n (in this case, the length parameter is specified to be $l(n) \geq 2n^3 + n$) and do not explicitly consider adaptive input and tag selection for the one left-session. But a closer investigation shows that the PRZK can be extended to work for tags of length $O(n)$ (and inputs of length $poly(n)$) with length parameter $l(n) \geq O(n^3)$ (the actual length parameter $l(n)$ is specific to the tag length), and for the more general case of adaptive left-session tag and input selection.

3 On CMIM with *Full* Adaptive Input Selection

In the traditional formulation of the CMIM settings (and also the stand-alone MIM settings), there are two levels of input-selecting capabilities for the CMIM adversary: (1) CMIM with predetermined left-session inputs, in which the inputs to left sessions are predetermined, and the CMIM adversary $\mathcal{A}$ can only set inputs to right sessions; (2) CMIM with adaptive input selection, in which $\mathcal{A}$ can set, adaptively based on its view, the inputs to both left sessions and right sessions. But, in the traditional formulation of CMIM, both for CMIM with predetermined left-session inputs and for CMIM with adaptive input selection, the CMIM adversary $\mathcal{A}$ is required (limited) to set the input of each session at the beginning of that session. We note that this requirement, on input selection in traditional CMIM formulation, could essentially limit the power of the CMIM adversary in certain natural settings. We give some concrete examples below.

Consider any protocol resulted from the composition of a coin-tossing protocol and a protocol in the CRS model. In most often cases, the input to the underlying protocol in the CRS model, denoted the CRS-protocol for notation simplicity, is also the input to the whole composed protocol. Note that the input to the underlying CRS-protocol can be set after the coin-tossing phase is finished, furthermore, can be set only at the last message of the composed protocol. We remark that it is true that for adaptive adversary in the CRS model, it is allowed to set statements based on the CRS. In other words, mandating the adversary to predetermine the input to the underlying CRS-protocol, without seeing the output of coin-tossing that serves as the underlying CRS, clearly limits the power of the adversary and thus weakens the provable security established for the composed protocol.

Another example is the Feige-Shamir-ZK-like protocol [33, 34], which consists of two sub-protocols (for presentation convenience, we call them verifier's sub-protocol and prover's sub-protocol) and the input of the protocol is only used in the prover's sub-protocol. The prover can set and prove the statements in the prover's sub-protocol, only after the verifier has successfully finished the verifier's sub-protocol in which the verifier proves some knowledge (e.g., its secret-key) to the prover. In this case, the adversary can take advantage of the verifier's sub-protocol interactions to set and prove inputs to the subsequent prover's sub-protocol, especially when the Feige-Shamir-ZK-like protocol is run *concurrently* in the *public-key* model [70]. Again, an adversary, as well as the honest prover, could set the input to a session only at the last message of the session, for example, considering the prover's sub-protocol is the Lapidot-Shamir WIPOK protocol [51]. As demonstrated in [70, 71, 73] and in this work, letting the adversary adaptively determine inputs, in view of the concurrent executions of the verifier's sub-protocol in the public-key model, renders strictly stronger power to the adversary.

In contrast, by CMIM with *full* adaptive input selection, we mean that a CMIM adversary can set inputs to both left sessions and right sessions; furthermore (and different from the traditional formulation of adaptive input selection), the adversary does not necessarily set the input to each session at the beginning of the session; Rather, the input may be set on the way of the session, and is based on the whole transcript evolution (among other concurrent sessions and the current session). Though the adversary is allowed to set inputs at any points of the concurrent execution evolution, whenever at some point the subsequent activities of an honest player in a session may utilize the input of the session while the adversary did not provide the input, the honest player just simply aborts the session. Similar to traditional CMIM with predetermined left-session inputs, we can define CMIM with predetermined left-session inputs but full adaptive input selection on the right, in which the inputs to left sessions are fixed and the CMIM adversary only sets inputs to right sessions in the above fully adaptive way.

From above clarifications, we conclude that allowing the CMIM adversary the capability of full adaptive input selection, in particular not necessarily predetermining the inputs of sessions at the start of each session, is a more natural formulation (and also more natural scenarios) for cryptographic protocols to be CNM-secure against CMIM adversaries with adaptive input selection. It renders stronger capability to the adversary, and thus allows us to achieve stronger provable CNM security. The general CNM feasibility in the BPK model established in this work is against CMIM with the capability of full adaptive input selection (and the capability of adaptive language selection for right sessions).

On achieving CNM with full adaptive input selection. We briefly note that no previous ZK protocols in the BPK model or the plain model were proved to be CNM-secure against even CMIM with predetermined left-session inputs but full adaptive input selection on the right (i.e., the inputs to left sessions are predetermined and the CMIM adversary only sets inputs to right sessions in the fully adaptive way), needless to say to be CNM secure against CMIM with full adaptive input selection. Specifically, the standard simulation-extraction paradigm for showing CNM security fails, in general, when the CMIM adversary is allowed the capability of full adaptive input selection.

In more detail, the standard simulation-extraction paradigm for establishing CNM security works as follows: the simulator first outputs an indistinguishable simulated transcript; and then extracts the witnesses to (different) inputs of successful right sessions appearing in the simulated transcript, *one by one sequentially*, by applying some assured underlying knowledge-extractor. This paradigm can work for CMIM adversary with the capability of traditional adaptive input selection, as the input to each right session is fixed at the beginning of the right session; Thus, applying knowledge-extractor on the right session does not change the statement of the session, which has appeared and is fixed in the simulated transcript.

But, for CMIM adversary of fully adaptive input selection, the standard simulation-extraction paradigm fails in general in this case. In particular, considering the adversary always sets inputs to right sessions only at the last message of each right session, such case applies to both of the above illustrated protocol examples: composing coin-tossing and NIZK, and the Feige-Shamir-ZK-like protocols. In this case, when we apply knowledge-extractor on a successful right session, the statement of this session will however also be changed, which means that the extractor may never extract the witness to the same statement appearing and being fixed in the simulated transcript.

4 Formulating CNMZK in the Public-Key Model, Revisited

In this section, we briefly re-examine, and clarify subtleties of, the formulation of CNMZK in the public-key model. More details are presented in Appendix D.

Traditional CNMZK formulation roughly is: for any PPT CMIM adversary $\mathcal{A}$, there exists a PPT simulator/extractor S such that: (1) S outputs a simulated transcript str indistinguishable from the real view of $\mathcal{A}$. (2) Given str, for any successful right session in str on a common input $\hat{x}$ different from those of left sessions, S will also output a valid $\mathcal{NP}$-witness of $\hat{x}$.

We highlight, motivated by concrete attacks against existing protocols in the BPK model, a key difference between CNM in the standard model and CNM in the public-key model, which traditional CNM formulation does not capture. For the CMIM setting in the standard model, honest verifiers are PPT algorithms. In this case, traditional CNM formulation only considers the extra advantages the CMIM can get from concurrent left sessions, as the actions of honest verifiers in right sessions can be efficiently emulated perfectly. But, for the CMIM setting in the public-key model, honest verifiers possess secret values (i.e, secret-keys) that can *not* be computed out efficiently from the public-keys. That is, for protocols in the public-key model, the CMIM adversary can get extra advantages both from the left sessions and *from the right sessions*. To emulate the actions of honest verifiers in the public-key model, we have to allow the simulator/extractor to simulate the key-generation stages of honest verifiers [73]. Put in other words, in its simulation/extration S actually takes the corresponding secret-keys of honest verifiers. In this case, as clarified in [73], knowledge-extraction does not guarantee that the CMIM adversary does indeed "know" the extracted witnesses to successful right sessions.

With the above key difference in mind, we investigate reformulating the CNM notion in the public-key model. Above all, besides requiring the ability of simulation/extraction, we need to mandate that for any CMIM-adversary the witnesses extracted for right sessions are "*independent*" of the secret-keys used by the simulator/extractor S (who emulates honest verifiers in the simulation/extraction). Such property is named concurrent non-malleable knowledge-extraction independence (CNMKEI). CNMKEI is formulated by extending the formulation of concurrent knowledge-extraction (CKE) of [73] into the more complicated CMIM setting (the CKE notion is formulated with adversaries only interacting with honest verifiers but without interacting with provers). Roughly, the CNMKEI is formulated as follows.

CNMKEI IN THE PUBLIC-KEY MODEL: We require that for any PPT CMIM-adversary $\mathcal{A}$ in the BPK model, there exists a PPT simulator/extractor S such that the following holds: $\Pr[\mathcal{R}(\widehat{W}, SK_V, str)] = 1$ is negligibly close to $\Pr[\mathcal{R}(\widehat{W}, SK'_V, str)] = 1$ for any polynomial-time computable relation $\mathcal{R}$, where SK'_V is some element *randomly and independently* distributed over the space of S_V, str is the simulated transcript indistinguishable from the real view of $\mathcal{A}$, and $\widehat{W}$ are the joint witnesses extracted to successful right sessions in str. Here, for any right session that is aborted (due to CMIM adversary abortion or verifier verification failure) or is of common input identical to that of one left session, the corresponding witness to that right session is set to be a special symbol $\perp$. We stress that, in the simulation-extraction, the simulator S only simulates the pair (PK_V, SK_V) (that is of identical distribution to that of the output of the key-generation stage of the honest verifier), it still uses the *same* real public-key PK_P of the honest prover without knowing the secret-key SK_P (which guarantees the concurrent ZK implication).

We also remind that our CNMKEI formulation implicitly assumes that verifier's public-key corresponds to multiple secret-keys, which however can typically be achieved with the common key-pair trick [63]. In Appendix D.3, we show that existing CNM formulations in the public-key model do not capture

CNMKEI. In general, cryptography literature should welcome diversified approaches for modeling and achieving security goals of cryptographic systems, particularly witnessed by the evolution history of public-key encryption. The possibility of achieving CNMZK with adaptive input selection in the BPK model turns out to be quite subtle, as it appears to be in conflict with Lindell's impossibility results on concurrent composition with adaptive input selection [56, 54]. A careful investigation shows CNMZK with *full* adaptive input selection is still possible in the BPK model. The reader is refer to Appendix D.3 for more details.

5 Constant-Round CNM Coin-Tossing in the BPK Model

Let $\langle L, R\rangle$ be a coin-tossing protocol between a left-player L and a right-player R. We abuse the notations L and R in this section. Specifically, L stands for the left-player and in some context we may explicitly indicates L to be a language, R stands for the right-player and in some context we may explicitly indicates R to be a relation.

To formulate CNMCT in the complex CMIM setting, the rough idea is: for any CMIM adversary $\mathcal{A}$ there exists a PPT simulator S such that: (1) S outputs a simulated transcript str indistinguishable from the real view of $\mathcal{A}$, together with some state information sta; (2) S can set, at its wish, "*random* coin-tossing outputs" for all (left and right) sessions in str, in the sense that S learns the corresponding trapdoor information (included in sta) of the coin-tossing output of each session. Intuitively, such formulation implies the traditional simulation-extraction CNM security. But, with the goal of transforming CNM cryptography from CRS model into the weaker BPK model in mind, some terms need to be further deliberated.

Above all, we need require the combination of str and sta should be independent of the secret-key emulated and used by the simulator. This is necessary to guarantee that $\mathcal{A}$ knows what it claims to know in its CMIM attack.

Secondly, we should mandate the ability of *online* setting coin-tossing outputs of all sessions appearing in str, in the sense that S sets the coin-tossing outputs and the corresponding trapdoor information (encoded in sta) in an online way at the same time of forming the str. This is critical to guarantee CNM security against CMIM with full adaptive input selection.

Finally, we need to make clear the meaning of "random coin-tossing outputs". One formulation is to require that all coin-tossing outputs are independent random strings. Such formalization rules out the natural copying strategy by definition, and thus is too strong to capture naturally secure protocols. On the other hand, in order to allow the copying strategy to the CMIM, an alternative relaxed formulation is to only require that the coin-tossing output of each *individual* session is random. But, this alternative formalization is too week to rule out naturally insecure protocols (for instance, consider that the CMIM manages to set the outputs of some sessions to be maliciously correlated and even to be identical). The right formulation should essentially be: *the coin-tossing output of each left (resp., right) session is either independent of the outputs of all other sessions OR copied from the output of one right (resp., left) session on the opposite CMIM part; furthermore, the output of each session in one CMIM part can be copied into the opposite CMIM part at most once.*

Legitimate CRS-simulating algorithm $\mathcal{M}_{CRS}$. Let $(r, \tau_r) \longleftarrow \mathcal{M}_{CRS}(1^n)$, where $\mathcal{M}_{CRS}$ is a PPT algorithm. The PPT algorithm $\mathcal{M}_{CRS}$ is called a legitimate CRS-simulating algorithm with respect to a polynomial-time computable CRS-trapdoorness validating relation $\mathcal{R}_{CRS}$, if the distribution of its first output, i.e., r, is computationally indistinguishable from U_n (the uniform distribution over strings of length n), and $\mathcal{R}_{CRS}(r, \tau_r) = 1$ for all outputs of $\mathcal{M}_{CRS}$ (typically, τ_r is some trapdoor information about r). For a positive polynomial $s(\cdot)$, we denote by $(\{r_1, r_2, \cdot, r_{s(n)}\}, \{\tau_{r_1}, \tau_{r_2}, \cdots, \tau_{r_{s(n)}}\}) \longleftarrow \mathcal{M}_{CRS}^{s(n)}(1^n)$ the output of the experiment of running $\mathcal{M}_{CRS}(1^n)$ *independently* $s(n)$ times, where for any i, $1 \leq i \leq s(n)$, (r_i, τ_{r_i}) denotes the output of the i-th independent execution of $\mathcal{M}_{CRS}$.

$\mathcal{M}_{CRS}$ trivially achievable distribution. Let G be a set of pairs of integers $\{(i_1, j_1), (i_2, j_2), \cdots, (i_t, j_t)\}$, where $1 \leq i_1 < i_2 < \cdots < i_t \leq s(n)$ and $1 \leq j_1, j_2, \cdots, j_t \leq s(n)$ are distinct integers, and $0 \leq t \leq s(n)$ such that G is defined to be the empty set when $t = 0$. Let $\mathcal{M}_{s,n,G}$ be the probability distribution over $(\{0,1\}^n)^{2s(n)}$, obtained by first generating $2s(n) - t$ n-bit strings $\{x_m, y_k | m \in \{1, 2, \cdots, s(n)\}, k \in \{1, 2, \cdots, s(n)\} - \{j_1, j_2, \cdots, j_t\}\}$, by running $\mathcal{M}(1^n)$ independently $2s(n) - t$ times, and then defining $y_{j_d} = x_{i_d}$ for $1 \leq d \leq t$ and taking $(x_1, x_2, \cdots, x_{s(n)}, y_1, y_2, \cdots, y_{s(n)})$ as the output. A probability distribution over $(\{0,1\}^n)^{2s(n)}$ is called *$\mathcal{M}$-trivially achievable*, if it is a convex combination of $U_{s,n,G}$ over all G's.

Definition 5.1 (concurrently non-malleable coin-tossing CNMCT) *Let $\Pi = \langle L, R \rangle$ be a two-party protocol in the BPK model, where $L = (L_{KEY}, L_{PROOF})$ and $R = (R_{KEY}, R_{PROOF})$. We say that Π is a concurrently non-malleable coin-tossing protocol in the BPK model w.r.t. some key-validating relations $\mathcal{R}^L_{KEY}$ and $\mathcal{R}^R_{KEY}$, if for any PPT $s(n)$-CMIM adversary $\mathcal{A}$ in the BPK model there exists a probabilistic (expected) polynomial-time algorithm $S = (S_{KEY}, S_{PROOF})$ such that, for any sufficiently large n, any auxiliary input $z \in \{0,1\}^*$, any PPT CRS-simulating algorithm $\mathcal{M}_{CRS}$ and any polynomial-time computable (CRS-trapdoor validating) relation $\mathcal{R}_{CRS}$, and any polynomial-time computable (SK-independence distinguishing) relation $\mathcal{R}$ (with components drawn from $\{0,1\}^* \cup \{\perp\}$), the following hold, in accordance with the experiment $\mathsf{Expt}_{\mathbf{CNMCT}}(1^n, z)$ described below:*

$\mathsf{Expt}_{\mathbf{CNMCT}}(1^n, z)$

Honest left-player key-generation:
$(PK_L, SK_L) \longleftarrow L_{KEY}(1^n)$. Denote by $\mathcal{K}_L$ the set of all legitimate public-keys generated by $L_{KEY}(1^n)$. *Note that the execution of L_{KEY} is independent from the simulation below. In particular, only the public-key PK_L is passed on to the simulator.*

The simulator $S = (S_{KEY}, S_{PROOF})$:
$(PK_R, SK_R, SK'_R) \longleftarrow S_{KEY}(1^n)$, where the distribution of (PK_R, SK_R) is identical with that of the output of the key-generation stage of the honest right-player R (i.e., R_{KEY}), $\mathcal{R}^R_{KEY}(PK_R, SK_R) = \mathcal{R}^R_{KEY}(PK_R, SK'_R) = 1$ and the distributions of SK_R and SK'_R are identical and *independent*.

$(str, sta) \longleftarrow S_{PROOF}^{\mathcal{A}(1^n,\ PK_L,\ PK_R,\ z)}(1^n, z, PK_L, PK_R, SK_R)$. That is, on inputs $(1^n, z, PK_L, PK_R, SK_R)$ and with oracle access to $\mathcal{A}(1^n, PK_L, PK_R, z)$ (by providing random coins to $\mathcal{A}$ and running $\mathcal{A}$ as a subroutine), the simulator S outputs a simulated transcript str and some state information sta. Denote by $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots, R_L^{(s(n))}\}$ the set of outputs of the $s(n)$ left sessions in str and by $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots, R_R^{(s(n))}\}$ the set of outputs of the $s(n)$ right sessions in str. The state information sta consists, among others, of two sub-sets (of $s(n)$ components each): $sta_L = \{sta_L^{(1)}, sta_L^{(2)}, \cdots, sta_L^{(s(n))}\}$ and $sta_R = \{sta_R^{(1)}, sta_R^{(2)}, \cdots, sta_R^{(s(n))})\}$. *Note that S does not know secret-key SK_L of honest left player, that is, S can emulate the honest left-player only from its public-key PK_L.*

For any $z \in \{0,1\}^*$, we denote by $S(1^n, z)$ the random variable str (in accordance with above processes of L_{KEY}, S_{KEY}, and S_{PROOF}). For any $z \in \{0,1\}^*$, any $PK_L \in \mathcal{K}_L$ and $(PK_R, SK_R) \in \mathcal{R}^R_{KEY}$, we denote by $S(1^n, z, PK_L, PK_R, SK_R)$ the random variable $S(1^n, z)$ specific to (PK_L, PK_R, SK_R).

- **Simulatability.** *The following ensembles are indistinguishable:* $\{S(1^n, z, PK_L, PK_R, SK_R)\}_{1^n, PK_L \in \mathcal{K}_L, (PK_R, SK_R) \in \mathcal{R}^R_{KEY}, z \in \{0,1\}^*}$ *and* $\{view_{\mathcal{A}}^{L(SK_L), R(SK_R)}(1^n, z, PK_L, PK_R)\}_{1^n, PK_L \in \mathcal{K}_L, (PK_R, SK_R) \in \mathcal{R}^R_{KEY}, z \in \{0,1\}^*}$ *(defined in accordance with the experiment $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, z)$ depicted in Section 2). This in particular implies that the probability ensembles $\{S(1^n, z)\}_{1^n, z \in \{0,1\}^*}$ and $\{view_{\mathcal{A}}(1^n, z)\}_{1^n, z \in \{0,1\}^*}$ are indistinguishable.*
- **Strategy-restricted and predefinable randomness.** *With overwhelming probability, both the distribution of (R_L, sta_L) and that of (R_R, sta_R) are identical to the distribution of $\mathcal{M}_{CRS}^{s(n)}(1^n)$; furthermore, the distribution of (R_L, R_R) is $\mathcal{M}$-trivially achievable.*

- **Secret-key independence.** $|\Pr[\mathcal{R}(SK_R, str, sta) = 1] - \Pr[\mathcal{R}(SK'_R, str, sta) = 1]|$ *is negligible.*

The probabilities are taken over the randomness of S in the key-generation stage (i.e., the randomness for generating (PK_R, SK_R, SK'_R)) and in all proof stages, the randomness of L_{KEY}, the randomness of $\mathcal{M}_{CRS}$, and the randomness of $\mathcal{A}$.

On the ability of online setting all coin-tossing outputs and its implication of CNM security against CMIM of full adaptive input selection. Note that in the above CNMCT formulation, the simulator S not only outputs a simulated transcript that is indistinguishable from the real view of the CMIM adversary, but also, S sets and controls, at the same time *in an online way*, the coin-tossing outputs of *all* left and right sessions in the simulated transcript (in the sense that S knows the corresponding trapdoor information of *all* the coin-tossing outputs appearing in the simulated transcript). This ability of S plays several essential roles: Firstly, setting the outputs of all CNMCT sessions (at its wish in an online way) is essential, in general, to transform CNM cryptography in the CRS model into the BPK model, as in the security formulation and analysis of CNM protocols in the CRS model the simulator does control and set all simulated CRS; Secondly, such ability of S is critical for obtaining CNM security against CMIM with full adaptive input selection, which is addressed below.

For more detailed clarifications about this issue, consider a protocol (e.g., a ZK protocol) that is resulted from the composition of a coin-tossing protocol in the BPK model and a protocol (e.g., an NIZK protocol) in the CRS model, and assume the CMIM adversary sets input to each session of the composed protocol at the last message of that session. In particular, the input to each session can be an arbitrary function of the coin-tossing output and will be different with respect to different coin-tossing outputs. Now, suppose the simulator/extractor cannot set the coin-tossing outputs of all right sessions *in an online way*; That is, for some (at least one) successful right sessions in the simulated transcript, the simulator fails in setting the coin-tossing outputs of these sessions, and thus learning no trapdoor information enabling on-line knowledge-extraction. In case the inputs of these right sessions do not appear as inputs of left sessions, then, in order to extract witnesses to the inputs of such successful right sessions appeared in the simulated transcript, the simulator/extractor has to rewind the CMIM adversary and manages to set, one by one sequentially, the coin-tossing outputs of these right sessions. But, the problem is: whenever the simulator/extractor is finally able to set (if it is possible), at its wish, the output of a right session in question, the input to that right session set by the CMIM adversary is however changed (as it is determined by the output of coin-tossing). This means that the simulator/extractor may never be able to extract the witnesses to all the inputs of successful right sessions appeared in the simulated transcript. The above arguments also apply to Feige-Shamir-ZK-like protocols as illustrated in Section 3. We remark that, it is the ability of online setting the outputs of all coin-tossing sessions, in our CNMCT formulation and security analysis, that enables us to obtain CNM security against CMIM of full adaptive input selection.

On the generality of CNMCT. We first note that CNMCT in the BPK model actually implies, or *serves as the basis to formulate*, concurrent non-malleability with full adaptive input selection for any cryptographic protocols in the BPK model. The reason is: concurrent non-malleability for any functionality can be implemented in the common random string model [21, 14]. By composing any concurrent non-malleable cryptographic protocol in the CRS model with a CNMCT protocol in the BPK model, with the output of CNMCT serving as the common random string of the underlying CNM-secure protocol in the CRS model, we can transform it into a CNM-secure protocol in the BPK model. In particular, we can view the composed protocol as a special (extended) coin-tossing protocol.

With CNMZK as an illustrative example, when composed with adaptive non-malleable NIZKAOK (e.g., the robust NIZK of [21] for $\mathcal{NP}$), CNMCT implies (tag-based) CNMZKAOK (for $\mathcal{NP}$) with full adaptive input selection in the BPK model. But, we do not need to explicitly formulate the (adaptive input-selecting) CNM security for ZK protocols in the BPK model. Specifically, we can view the composed protocol (of CNMCT and robust NIZK) as a special version of coin-tossing and note that in this case (str, τ) implies knowledge-extraction. Then, the properties of simulatability and strategy-restricted and predetermined randomness of CNMCT implies simulation-extraction, by viewing $\mathcal{M}_{CRS}$

as the CRS simulator of the underlying NMNIZK. The secret-key independent knowledge extraction is derived from the property of secret-key independence of CNMCT.

5.1 Implementation and analysis of constant-round CNMCT in the BPK model

High-level overview of the CNMCT implementation. We design a coin-tossing mechanism in the BPK model, which allows each player to set the coin-tossing output whenever it learns its peers's secret-key. The starting point is the basic Blum-Lindell coin-tossing [9, 52]: the left-player L commits a random string σ, using randomness s_σ, to $c = C(\sigma, s_\sigma)$ with a statistically-binding commitment scheme C; The right-player R responds with a random string r_r; L sends back $r = \sigma \oplus r_l$ and proves the knowledge of (σ, s_σ). To render the simulator the ability of online setting coin-tossing outputs against malicious right-players, R proves its knowledge of its secret-key SK_R (using the key-pair trick of [63]), and L accordingly proves the knowledge of either (σ, s_σ) or SK_R. To render the ability of online setting coin-tossing outputs against malicious left-players, L registers $c = C(\sigma, s_\sigma)$ as its public-key and treats σ as the seed of a pseudorandom function PRF; L then sends r'_l that commits to $r_l = PRF_\sigma(r'_l)$; after receiving r_r from R, it returns back $r = r_l \oplus r_r$ and proves the knowledge of either its secret-key $SK_L = (\sigma, s_\sigma)$ (such that $r = r_r \oplus PRF_\sigma(r'_l)$) or the right-player's secret-key SK_R. The underlying proof of knowledge is implemented with PRZK. But, *correct* knowledge-extraction with bare public-keys in the complex CMIM setting is quite subtle. At a very high level, the correct knowledge extraction, as well as the CNM security, is reduced to the one-left-many-right non-malleability of PRZK.

The constant-round CNMCT protocol $\langle L, R\rangle$ in the BPK model, is depicted in Figure 1 (page 10). Here, for presentation simplicity, we often write L and R to denote the left and right players directly without explicitly indicating the key-generation algorithm and the proof algorithm (which are implicitly clear from the context). Note that the PRZK is used as a building tool and is composed with other sub-protocols in the protocol, and that the left-tag of PRZK in Stage-5 is set interactively. The actual statements to be proved by commit-then-PRZK and PRZK (in Stage-1 and Stage-5) are got by applying $\mathcal{NP}$-reductions, while the tags remaining the same. Note that the tags of the underlying PRZK in Stage-1 and Stage-5 can be equal, both of which are $O(n)$ as both the OWF f and the statistically-binding commitment scheme C are required to be linear. For PRZK to work with the CNMCT construction, we should require the length parameter $l(n) \geq O(n^3)$ which are specific to the underlying tools f and C (in particular, $l(n) = n^4$ suffices for this work).

Theorem 5.1 *Under linear OWF, (non-interactive) linear statistically-binding commitments, and PRZK, the protocol* $\Pi = \langle L, R\rangle$ *depicted in Figure 1 is a constant-round CNMCT protocol in the BPK model.*

The proof details of Theorem 5.1 is presented in Appendix E. We present the analysis outline below.

The (high-level) description of the CNM simulation. For any s-CMIM adversary $\mathcal{A}$ in the BPK model, consider a mental simulator M who, on input $(1^n, z, PK_L, PK_R, SK_R, F')$, additionally knows secret-keys corresponding to all public-keys registered by $\mathcal{A}$ in F'. For any i, $1 \leq i \leq s(n)$, in the i-th left-session w.r.t. a (right-player) public-key $PK_R^{(j)} \in F = F' \cup \{PK_L, PK_R\}$, the Stage-4 message $r^{(i)}$ and the state-information are set to be $(S_L^{(i)}, \tau_L^{(i)})$ by running the CRS-simulating algorithm $\mathcal{M}_{CRS}(1^n)$; then M commits the secret-key $SK_R^{(j)}$ (assumed known to it) to $c_{crs}^{(i)}$ and finishes the PRZK with $SK_R^{(j)}$ as the witness in Stage-5. For the i-th right-session w.r.t. $PK_L^{(j)}$ (with $SK_L^{(j)} = (\sigma^{(j)}, s_\sigma^{(j)})$), after receiving Stage-2 message $\tilde{r}_l^{(i)\prime}$, M runs $\mathcal{M}_{CRS}(1^n)$ to get the output denoted $(S_R^{(i)}, \tau_R^{(i)})$, sends $r_r^{(i)} = PRF_{\sigma^{(j)}}(\tilde{r}_l^{(i)\prime}) \oplus S_R^{(i)}$ at Stage-3. Here, the notation of m denotes a message sent by the simulator (emulating honest players), and $\tilde{m}$ denotes the arbitrary message sent by $\mathcal{A}$. To build up the simulator S from scratch, we resort to the key-coverage techniques of [12, 5, 41]. Specifically, $S(s_b)$ with simulated $SK_R = s_b$, works in at most $s(n) + 1$ repetitions. In each simulation repetition, it either successfully finishes the simulation or "covers" a new public-key. But, key-coverage in the complex CMIM setting with bare public-keys turns out to be much more complicated and subtler.

The CNM simulation is described in Figure 2 (page 34). Note that in Case-R2 of right-session simulation w.r.t. the uncovered left-player key $PK_L^{(j)} = PK_L$, S does not try to extract the secret-key

Right-player key registration: Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a linear one-way function. On a security parameter n, the right-player R (actually R_{KEY}) randomly selects s_0, s_1 from $\{0,1\}^n$, computes $y_0 = f(s_0)$, $y_1 = f(s_1)$. R publishes $PK_R = (y_0, y_1)$ as its public-key, and keeps $SK_R = s_b$ as its secret-key for a random bit $b \in \{0,1\}$ while discarding $SK' = s_{1-b}$. Define $\mathcal{R}^R_{KEY} = \{((y_0, y_1), x) | y_0 = f(x) \vee y_1 = f(x)\}$, and $\mathcal{K}_R$ the corresponding $\mathcal{NP}$-language.

Left-player key registration: Let C be a non-interactive statistically-binding *linear* commitment scheme. Each left-player L (actually L_{KEY}) selects $\sigma \in \{0,1\}^n$ and $s_\sigma \in \{0,1\}^{poly(n)}$ uniformly at random, computes $c = C(\sigma, s_\sigma)$ (i.e., committing to σ using randomness s_σ). Set $PK_L = c$ and $SK_L = (\sigma, s_\sigma)$, where σ serves as the random seed of a pseudorandom function PRF. Define $\mathcal{K}_L = \{c | \exists (x, s)\ s.\ t.\ \ c = C(x, s)\}$.

Stage-1. The right-player R (actually R_{PROOF}) computes and sends $c_{sk} = C(SK_R, s_{sk})$, where C is the statistically-binding commitment scheme and s_{sk} is the randomness used for commitment; Define $\mathcal{L}_{SK} = \{((y_0, y_1), c_{sk}) | \exists (s_{sk}, SK)\ s.t.\ c_{sk} = C(SK, s_{sk}) \wedge (y_0 = f(SK) \vee y_1 = f(SK))\}$. Then, R proves to the left-player L the knowledge of (SK_R, s_{sk}) such that $((PK_R, c_{sk}), (SK_R, c_{sk})) \in \mathcal{R}_{\mathcal{L}_{SK}}$, by running the Pass-Rosen non-malleable ZK (PRZK) for $\mathcal{NP}$ with the tag set to be $(PK_L, PK_R = (y_0, y_1))$ that is referred to as the *right* tag. The composed protocol of statistically-binding commitment and PRZK is called *commit-then-PRZK*.

Stage-2. The left player L (actually L_{PROOF}) randomly selects $r'_l \leftarrow \{0,1\}^n$, and sends r'_l to R.

Stage-3. The right player R randomly selects $r_r \leftarrow \{0,1\}^n$ and sends r_r to the left player.

Stage-4. The left player computes $r_l = PRF_\sigma(r'_l)$ (where σ is the random seed of PRF committed in L's public-key PK_L), and sends $r = r_l \oplus r_r$ to the right player.

Stage-5. L computes and sends $c_{crs} = C(\sigma || s_\sigma, s_{crs})$, where "$||$" denotes the operation of string concatenation. Define $\mathcal{L}_{CRS} = \{(PK_L = C(\sigma, s_\sigma), PK_R = (y_0, y_1), r'_l, r_r, r, c_{crs}) | \exists (x, s, s_{crs})\ s.t.\ c_{crs} = C(x || s, s_{crs}) \wedge [(PK_L = C(x, s) \wedge PRF_x(r'_l) = r \oplus r_r) \vee y_0 = f(x) \vee y_1 = f(x)]\}$. Then, L proves to R the knowledge $(\sigma, s_\sigma, s_{crs})$ such that $((PK_L, PK_R, r'_l, r_r, r, c_{crs}), (\sigma, s_\sigma, s_{crs})) \in \mathcal{R}_{\mathcal{L}_{CRS}}$, by running the PRZK for $\mathcal{NP}$ with the tag set to be (PK_L, r_r, r) that is referred to as the *left* tag. That is, L proves to R that either the value committed in c_{crs} is $SK_L = (\sigma, s_\sigma)$ such that $PRF_\sigma(r'_l) = r \oplus r_r$ OR the n-bit prefix of the committed value is the preimage of either y_0 or y_1. W.l.o.g., we can assume the left-tag (PK_L, r_r, r) and the right-tag (PK_L, y_0, y_1) are of the same length (e.g., f is simply a one-way permutation).

The result of the protocol is the string r. We will use the convention that if one of the parties aborts (or fails to provide a valid proof) then the other party determines the result of the protocol.

Figure 1: Constant-round CNMCT in the BPK model

of PK_L. In the following analysis, we show that in this case, with overwhelming probability, the tag of Stage-5 of this successful right session is identical to that of Stage-5 of a left-session. As the tag of Stage-5 of a session consists of the session output (i.e., the coin-tossing output), this implies that the session output of the right-session is identical to that of one of left-sessions. Moreover, we show that with overwhelming probability each left-session output can appear, as session output, in at most one successful right-session. In the unlikely event that $\mathcal{A}$ finishes a right session and the Stage-1 of a left-session simultaneously, both of which are w.r.t. uncovered public-keys, extracting SK_R in left simulation part takes priority.

Simulatability. Assuming truly random first output of $\mathcal{M}_{CRS}$ (the analysis to the pseudorandom case is direct), there are two differences between the output of the mental simulator M and the real view of $\mathcal{A}$: (1) Truly random (in simulation) vs. pseudorandom (in real execution) Stage-4 messages of left-sessions. The distinguishable gap caused by such difference can be ruled out, using hybrid arguments, by the pseudorandomness of PRF and the hiding property of PK_L that commits to the seed of PRF; (2) Witness difference in Stage-5 of left sessions: M always uses the (right-player) secret-key $SK_R^{(j)}$, while the honest left-player L always uses SK_L in real execution. The second difference can be ruled out, using hybrid arguments, by the regular WI property of commit-then-PRZK. For the simulator $S(s_b)$ from scratch with key-coverage, the subtle point here is: the value extracted from successful Stage-5

External honest left-player key-generation: Let $(PK_L, SK_L) \longleftarrow L_{KEY}(1^n)$, where $PK_L = c$ and $SK_L = (\sigma, s_\sigma)$ such that $\sigma \in \{0,1\}^n$ and $s_\sigma \in \{0,1\}^{t(n)}$ and $c = C(\sigma, s_\sigma)$. This captures the fact that S does not know SK_L and can emulate the honest left-player with the same public-key PK_L.

Public-key file generation:
$S_{KEY}(1^n)$ perfectly emulates the key-generation stage of the honest right-player, getting $PK_R = (y_0 = f(s_0), y_1 = f(s_1))$ and $SK_R = s_b$ and $SK'_R = s_{1-b}$ for a random bit b. Then, S_{KEY} runs $\mathcal{A}(1^n, PK_L, PK_R, z)$ to get (F', τ), where F' is a set of at most $s(n)$ public-keys and τ is the state information to be used by the proof stage of $\mathcal{A}$. The public-key file to be used in the proof-stage is $F = F' \cup \{PK_L, PK_R\}$.

$\mathcal{S} \leftarrow \{(PK_R, SK_R)\}$ (i.e. initiate the set of covered keys $\mathcal{S}$ to be $\{(PK_R, SK_R)\}$).
On input $(1^n, F', PK_L, PK_R, SK_R, \tau)$ and running $\mathcal{A}(PK_L, PK_R, F', \tau)$ as a subroutine, the following process is run by S_{PROOF} repeatedly at most $s(n)+1$ times. In each simulation repetition, S uses fresh randomness and tries to either end with a successful simulation or cover a new public-key in $F-\mathcal{S}$.

Straight-line left simulation:
In the i-th left concurrent session (ordered by the time-step in which the first round of each left-session is played) between S and $\mathcal{A}$ in the left CMIM interaction part w.r.t. a public-key $PK_R^{(j)} = (y_0^{(j)}, y_1^{(j)}) \in \mathcal{K}_R$, $1 \le i, j \le s(n)$, S acts as follows:

In case $\mathcal{A}$ successfully finishes Stage-1 and $PK_R^{(j)} \in F' - \mathcal{S}$, the simulator ends the current repetition of simulation trial, and starts to extract a secret-key $SK_R^{(j)}$ such that $\mathcal{R}^R_{KEY}(PK_R^{(j)}, SK_R^{(j)}) = 1$, which is guaranteed by the AOK property of PRZK. Then, let $\mathcal{S} \leftarrow \mathcal{S} \cup \{(PK_R^{(j)}, SK_R^{(j)})\}$, and move to the next repetition (with the accumulated covered-key set $\mathcal{S}$ and the same public-key file F).

In case $\mathcal{A}$ successfully finishes Stage-1 and $PK_R^{(j)} \in \mathcal{S}$ (i.e., S has already learnt the secret-key $SK_R^{(j)}$), S randomly selects $r_l^{(i)\prime} \leftarrow \{0,1\}^n$ and sends $r_l^{(i)\prime}$ to $\mathcal{A}$ at Stage-2. After receiving Stage-3 message, denoted $\tilde{r}_r^{(i)}$, from $\mathcal{A}$, S invokes $\mathcal{M}_{CRS}(1^n)$ and gets the output denoted $(S_L^{(i)}, \tau_L^{(i)})$. S then sends $r^{(i)} = S_L^{(i)}$ as the Stage-4 message (rather than sending back $r^{(i)} = PRF_\sigma(r_l^{(i)\prime}) \oplus \tilde{r}_r^{(i)}$ as the honest left-player does), and sets $sta_L^{(i)} = \tau_L^{(i)}$. In Stage-5, S computes and sends $c_{crs}^{(i)} = C(SK_R^{(j)} || 0^{t(n)}, s_{crs}^{(i)})$ to $\mathcal{A}$ (rather than sending back $c_{crs}^{(i)} = C(\sigma || s_\sigma)$ as the honest left-player does), where $t(n)$ is the length of s_σ in SK_L. Finally, S finishes the PRZK of Stage-5 with $(SK_R^{(j)}, s_{crs}^{(i)})$ as its witness and $(PK_L, \tilde{r}_r^{(i)}, S_L^{(i)})$ as the tag.

Straight-line right simulation:
In the i-th right concurrent session (ordered by the time-step in which the first round of each right-session is played) between S and $\mathcal{A}$ in the right CMIM interaction part with respect to a public-key $PK_L^{(j)} = c^{(j)} \in \mathcal{K}_L$, $1 \le i, j \le s(n)$, S acts as follows:

S perfectly emulates honest right-player in Stage-1 of any right session, with SK_R as the witness to commit-then-PRZK and $(PK_L^{(j)}, PK_R)$ as the tag.

Case-R1: If $PK_L^{(j)} \in \mathcal{S}$ (i.e., S has already learnt the secret-key $SK_L^{(j)} = (\sigma^{(j)}, s_\sigma^{(j)})$), after receiving $\tilde{r}_l^{(i)\prime}$ from $\mathcal{A}$ at Stage-2, S runs $\mathcal{M}_{CRS}(1^n)$ and gets the output denoted $(S_R^{(i)}, \tau_R^{(i)})$, and then computes and sends $PRF_{\sigma^{(j)}}(\tilde{r}_l^{(i)\prime}) \oplus S_R^{(i)}$ as Stage-3 message, and goes further as the honest right-player does.

Case-R2: If $PK_L^{(j)} \notin \mathcal{S} \cup \{PK_L\}$, and $\mathcal{A}$ successfully finishes the i-th right session (in which S just perfectly emulates the honest right-player of PK_R), then the simulator S ends the current repetition of simulation trial, and starts to extract a secret-key $SK_L^{(j)}$ such that $\mathcal{R}^L_{KEY}(PK_L^{(j)}, SK_L^{(j)}) = 1$. *In case S fails to extract such $SK_L^{(j)}$, S stops the simulation, and outputs a special symbol $\perp$ indicating simulation failure. Such simulation failure is called Case-R2 failure.* In case S successfully extracts such $SK_L^{(j)}$, then let $\mathcal{S} \leftarrow \mathcal{S} \cup \{(PK_L^{(j)}, SK_L^{(j)})\}$, and move to the next repetition. If $PK_L^{(j)} = PK_L$, S just works as the honest right-player does.

Setting sta_R: For successful i-th right session, if the Stage-4 message $\tilde{r}^{(i)}$ is $S_R^{(i)}$ or $S_L^{(k)}$ for some k, $1 \le k \le s(n)$, then $sta_R^{(i)}$ is set accordingly to $\tau_R^{(i)}$ or $\tau_L^{(k)}$; otherwise, $sta_R^{(i)}$ is set to be $\perp$.

Figure 2: The CNM simulation

of a right session w.r.t. $PK_L^{(j)}$, by the argument of knowledge (AOK) of PRZK, may not necessarily be $SK_L^{(j)}$, but may possibly be the preimage of $y_b = f(s_b)$ (due to the one-wayness of y_{1-b}, the value extracted cannot be the preimage of y_{1-b}). This is called *key-coverage failure* (i.e., the Case-R2 failure

Figure 3). All left is to show key-coverage failure occurs with negligible probability.

We first present some observations on commit-then-PRZK *with restricted input selection and indistinguishable auxiliary information*. Consider the following experiments: $\mathrm{EXPT}(1^n, w^b, aux^b)$, where $w^b \in \{0,1\}^n$ for $b \in \{0,1\}$. In $\mathrm{EXPT}(1^n, w^b, aux^b)$, the commit-then-PRZK for $\mathcal{NP}$ is run concurrently, and an m-CMIM adversary $\mathcal{A}$ for some polynomial $m(\cdot)$, possessing auxiliary information aux^b, can set the inputs and tags to prover instances in left sessions with the restriction: for any x_i, $1 \le i \le m(n)$, set by $\mathcal{A}$ for the i-th left session, the fixed value w^b is always a valid $\mathcal{NP}$-witness. Denote by $trans^b$ the transcript of the experiment $\mathrm{EXPT}(1^n, w^b, aux^b)$, and by $\widehat{W}^b = \{\hat{w}_1^b, \cdots, \hat{w}_{s(n)}^b\}$ the witnesses encoded (determined) by the statistically-binding commitments (at the beginning) of *successful* right sessions (of the commit-then-PRZK) in $trans^b$ *with tags different from those of left-sessions*. By a series of hybrid arguments, we can get: if $\{aux^0\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ and $\{aux^1\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ are indistinguishable, the ensembles $\{(trans^0, \widehat{W}^0)\}$ and $\{(trans^1, \widehat{W}^1)\}$, indexed by $\{n \in N, w^0 \in \{0,1\}^n, w^1 \in \{0,1\}^n\}$, are also indistinguishable.

Denote by $\mathcal{C}_b^k$ the set of covered key-pairs, corresponding to public-keys in $F - \{PK_R\}$, which is used by $S(1^n, s_b)$ in its k-th simulation repetition. Note that $\mathcal{C}_b$ does not include the simulated (PK_R, SK_R) now. The key observation here is: by viewing the messages involving $SK_R = s_b$ from S in the simulation (in Stage-1 of right sessions, or Stage-5 of left sessions in case $\mathcal{A}$ impersonates the honest right-player of PK_R) as from the instances of the prover $P(1^n, s_b)$ of commit-then-PRZK, the k-th simulation repetition actually amounts to the experiment of $\mathrm{EXPT}(1^n, w^b, aux^b)$, with w^b set to be s_b and aux^b set to be $\mathcal{C}_b^k$. By inductive steps, we can get $\{\mathcal{C}_0^k\}_{n,s_0,s_1}$ and $\{\mathcal{C}_1^k\}_{n,s_0,s_1}$ are indistinguishable, for any k, $1 \le k \le s(n)+1$. Suppose key-coverage failure occurs in the successful i-th right session w.r.t. an uncovered $PK_L^{(j)}$ during the k-th simulation repetition, by the tag-setting mechanism, the Stage-5 tag used by $\mathcal{A}$ in the i-th right session must be *distinct* (i.e., different from all tags used by the simulator for Stage-1 of right sessions and Stage-5 of left sessions). This means that the value committed to $\tilde{c}_{crs}^{(i)}$, and extracted efficiently, cannot be the preimage of y_b, from which key-coverage failure is ruled out.

Secret-key independence. For any pair (s_0, s_1) in the (simulated right-player) key-generation stage, denote by (str^b, sta^b) the output of $S(1^n, s_b)$ with $SK_R = s_b$. Suppose the secret-key independence property does not hold, there must exist a bit $\alpha \in \{0,1\}$ such that the difference between $\Pr[\mathcal{R}(s_\alpha, str^0, sta^0) = 1 | S \text{ uses } s_0 \text{ in generating } (str^0, sta^0)]$ and $\Pr[\mathcal{R}(s_\alpha, str^1, sta^1) = 1 | S \text{ uses } s_1 \text{ in generating } (str^1, sta^1)]$ is non-negligible. This implies (s_α, str^0, sta^0) and (s_α, str^1, sta^1) are distinguishable. But, note that the preceding analysis has already established that the ensembles $\{(str^0, sta^0)\}$ and $\{(str^1, sta^1)\}$, indexed by $\{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n\}$, are indistinguishable.

Strategy-restricted and predefinable randomness. This is essentially to show, with overwhelming probability, for any i, the output of the successful i-th right session w.r.t. $PK_L^{(j)}$ is either $S_R^{(i)}$ or $S_L^{(k)}$ for some k, $1 \le i, k \le s(n)$; furthermore, any left-session output $S_L^{(k)}$ can be the output for at most *one* successful right session.

As key-coverage failure occurs with negligible probability, we get $PK_L^{(j)} \in \mathcal{C}_b \cup \{PK_R, PK_L\}$, where $\mathcal{C}_b$ is the set of extracted-keys (corresponding to public-keys in $F - \{PK_R\}$) used by $S(s_b)$ in its last simulation repetition.

If $PK_L^{(j)} = PK_L$, the Stage-5 tag of the successful i-th right session must be identical to that of Stage-5 of a left session, which means the coin-tossing output is identical to the output of the left-session (note that each Stage-5 tag contains coin-tossing output of the session). Otherwise (i.e., the Stage-5 tag of the i-th right session is different from Stage-5 tags of all left-sessions), the Stage-5 tag of the i-th right session is *distinct*, which violates one of the following (by considering the possibilities of the value committed to $\tilde{c}_{crs}^{(i)}$ that can be efficiently extracted): one-wayness of PK_L (note S never uses SK_L in simulation), one-wayness of y_{1-b}, the one-left-many-right non-malleability of PRZK; For the case of $PK_L^{(j)} \neq PK_L$ (recall Stage-5 tags of left sessions always include PK_L), similar analysis shows the coin-tossing output is either $S_R^{(i)}$ or $S_L^{(k)}$ for some k.

Finally, suppose there are two successful right sessions of the same left-session output $S_L^{(k)}$, one of the two sessions, referred to as the i_b-th session, must be of *distinct* Stage-5 tag, which implies the public-key $PK_L^{(j)}$ used by $\mathcal{A}$ in that session is covered and is *not* PK_L (as any right-session w.r.t. PK_L is of a tag identical to that of one left-session). For the value committed to $\tilde{c}_{crs}^{(i_b)}$ (at the beginning of Stage-5 of the i_b-th right session), we can show it is neither $SK_L^{(j)}$ (as, otherwise, the $\mathcal{NP}$-statement successfully proved by PRZK in the Stage-5 of the i_b-th right session is actually false) nor the preimage of y_{1-b} (due to the one-wayness of f); Also, the value committed to $\tilde{c}_{crs}^{(i_b)}$ cannot be the preimage of y_b (by the one-left-many-right non-malleability of PRZK). Contradiction is reached in either case.

Acknowledgments. The third author thanks Rafael Pass and Alon Rosen for many helpful discussions.

References

[1] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *IEEE Symposium on Foundations of Computer Science*, pages 106-115, 2001.

[2] B. Barak. Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In *IEEE Symposium on Foundations of Computer Science*, pages , 2002.

[3] B. Barak, R. Canetti, J. B. Nielsen and R. Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *IEEE Symposium on Foundations of Computer Science*, pages 186-195, 2004.

[4] B. Barak and O. Goldreich. Universal Arguments and Their Applications. In*IEEE Conference on Computational Complexity*, pages 194-203, 2002.

[5] B. Barak, O. Goldreich, S. Goldwasser and Y. Lindell. Resettably-Sound Zero-Knowledge and Its Applications. In *IEEE Symposium on Foundations of Computer Science*, pages 116-125, 2001.

[6] B. Barak, M. Prabhakaran and A. Sahai. Concurrent Non-Malleable Zero-Knowledge In *IEEE Symposium on Foundations of Computer Science*, 2006.

[7] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1992, LNCS 740*, pages 390-420. Springer-Verlag, 1992.

[8] M. Bellare and O. Goldreich. On Probabilistic versus Deterministic Provers in the Definition of Proofs Of Knowledge. Electronic Colloquium on Computational Complexity, 13(136), 2006.

[9] M. Blum. Coin Flipping by Telephone. In *proc. IEEE Spring COMPCOM*, pages 133-137, 1982.

[10] M. Blum. How to Prove a Theorem so No One Else can Claim It. In Proceedings of the International Congress of Mathematicians, Berkeley, California, USA, 1986, pp. 1444-1451.

[11] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136-145, 2001.

[12] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 235-244, 2000.

[13] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(log\ n)$ Rounds. In *ACM Symposium on Theory of Computing*, pages 570-579, 2001.

[14] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *ACM Symposium on Theory of Computing*, pages 494-503, 2002.

[15] R. Cramer, I. Damgard and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Y. Desmedt (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1994, LNCS 839*, pages 174-187. Springer-Verlag, 1994.

[16] I. Damgård. On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs. In *G. Brassard (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1989, LNCS 435*, pages 17-27. Springer-Verlag, 1989.

[17] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *B. Preneel (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2000, LNCS 1807*, pages 418-430. Springer-Verlag, 2000.

[18] I. Damgard. On Σ-protocols. A lecture note for the course of Cryptographic Protocol Theory at Aarhus University, 2003. Available from: http://www.daimi.au.dk/∼ivan/CPT.html

[19] I. Damgard and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *ACM Symposium on Theory of Computing*, pages 426-437, 2003.

[20] I. Damgard, T. Pedersen and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. *Journal of Cryptology*, 10(3): 163-194, 1997. Preliminary version appears in Crypto 1993.

[21] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-Interactive Zero-Knowledge. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 566-598. Springer-Verlag, 2001.

[22] Y. Deng, G. Di Crescenzo, and D. Lin. Concurrently Non-Malleable Zero-Knowledge in the Authenticated Public-Key Model. Cryptology ePrint Archive, Report No. 2006/314, September 12, 2006.

[23] G. Di Crescenzo and I. Visconti. Concurrent Zero-Knowledge in the Public-Key Model. In *L. Caires et al. (Ed.): ICALP 2005, LNCS 3580*, pages 816-827. Springer-Verlag, 2005.

[24] G. Di Crescenzo and I. Visconti. On Defining Proofs of Knowledge in the Bare Public-Key Model. In *ICTCS*, 2007.

[25] G. Di Crescenzo, Y. Ishai and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *ACM Symposium on Theory of Computing*, pages 141-150, 1998.

[26] G. Di Crescenzo, J. Katz, R. Ostrovsky and A. Smith. Efficient and Non-Interactive Non-Malleable Commitments. In *B. Pfitzmann (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, pages 40-59. Springer-Verlag, 2001.

[27] G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Pre-Processing. In *M. J. Wiener (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1999, LNCS 1666*, pages 485-502. Springer-Verlag, 1999.

[28] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. In *ACM Symposium on Theory of Computing*, pages 542-552, 1991.

[29] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 409-418, 1998.

[30] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *H. Krawczyk (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1998, LNCS 1462*, pages 442-457. Springer-Verlag, 1998.

[31] T. El Gamal. A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31: 469-472, 1985.

[32] U. Feige. Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990. Available from: `http://www.wisdom.weizmann.ac.il/~feige`.

[33] U. Feige and Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *G. Brassard (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1989, LNCS 435*, pages 526-544. Springer-Verlag, 1989.

[34] U. Feige. Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D Thesis, Weizmann Institute of Science, 1990.

[35] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *ACM Symposium on Theory of Computing*, pages 416-426, 1990.

[36] U.Feige, D. Lapidot and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, 29(1): 1-28, 1999.

[37] M. Fischlin and R. Fischlin. Efficient Non-Malleable Commitment Schemes. In *M. Bellare (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2000, LNCS 1880*, pages 413-431. Springer-Verlag, 2000.

[38] O. Goldreich. *Foundation of Cryptography-Basic Tools*. Cambridge University Press, 2001.

[39] O. Goldreich. *Foundations of Cryptography-Basic Applications*. Cambridge University Press, 2002.

[40] O. Goldreich, S. Goldwasser and S. Micali. How to Construct Random Functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[41] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for $\mathcal{NP}$. *Journal of Cryptology*, 9(2): 167-189, 1996.

[42] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All language in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery*, 38(1): 691-729, 1991.

[43] S. Goldwasser, S. Micali and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks. *SIAM Journal on Computing*, 17(2): 281-308, 1988.

[44] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof System. *SIAM Journal on Computing*, 18(1): 186-208, 1989.

[45] I. Haitner and O. Reingold. Statistically-Hiding Commitment from Any One-Way Function. Cryptology ePrint Archive, Report No. 2006/436.

[46] I. Haitner, O. Horvitz, J. Katz, C. Koo, R. Morselli and R. Shaltiel. Reducing Complexity Assumptions for Statistically-Hiding Commitments. In *R. Cramer (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2005, LNCS 3494*, pages 58-77. Springer-Verlag, 2005.

[47] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes From Collision-Free Hashing. In *N. Koblitz (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, pages 201-215. Springer-Verlag, 1996.

[48] J. Håstad, R. Impagliazzo, L. A. Levin and M. Luby. Construction of a Pseudorandom Generator from Any One-Way Function. *SIAM Journal on Computing*, 28(4): 1364-1396, 1999.

[49] Y. T. Kalai, Y. Lindell and M. Prabhakaran. Concurrent Composition of Secure Protocols in the Timing Model. In *ACM Symposium on Theory of Computing*, pages 644-653, 2005.

[50] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656*, pages 578-595, Springer-Verlag, 2003.

[51] D. Lapidot and A. Shamir. Publicly-Verifiable Non-Interactive Zero-Knowledge Proofs. In *A.J. Menezes and S. A. Vanstone (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1990, LNCS 537*, pages 353-365. Springer-Verlag, 1990.

[52] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology*, 16(3): 143-184, 2003.

[53] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. In *ACM Symposium on Theory of Computing*, pages 683-692, 2003.

[54] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *IEEE Symposium on Foundations of Computer Science*, pages 394-403, 2003.

[55] Y. Lindell. Lower Bounds for Concurrent Self Composition. In Theory of Cryptography (TCC) 2004, LNCS 2951, pages 203-222, Springer-Verlag, 2004.

[56] Y. Lindell. Lower Bounds and Impossibility Results for Concurrenet Self Composition. *Journal of Cryptology*, to appear. Preliminary versions appear in [53] and [55].

[57] D. Micciancio and E. Petrank. Simulatable Commitments and Efficient Concurrent Zero-Knowledge. In *E. Biham (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2003, LNCS 2656* , pages 140-159. Springer-Verlag, 2003.

[58] S. Micali, R. pass and A. Rosen. Input-Indistinguishable Computation. In *IEEE Symposium on Foundations of Computer Science*, pages 3136-145, 2006.

[59] S. Micali and L. Reyzin. Soundness in the Public-Key Model. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 542–565. Springer-Verlag, 2001.

[60] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology*, 4(2): 151-158, 1991.

[61] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Perfect Zero-Knowledge Arguments for NP Using Any One-Way Permutation. *Journal of Cryptology*, 11(2): 87-108, 1998.

[62] M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 1(2): 231-262 (2004).

[63] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *ACM Symposium on Theory of Computing*, pages 427-437, 1990.

[64] R. Ostrovsky, G. Persiano and I. Visconti. Concurrent Non-Malleable Witness Indistinguishability and Its Applications. Electronic Colloquium on Computational Complexity, 13(95), 2006.

[65] R. Ostrovsky, G. Persiano and I. Visconti. Constant-Round Concurrent NMWI and Its Relation to NMZK. Revised version of [64], ECCC, March 2007.

[66] R. Pass. Personal communications, 2006.

[67] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *ACM Symposium on Theory of Computing*, pages 533-542, 2005.

[68] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. In *IEEE Symposium on Foundations of Computer Science*, pages 563-572, 2005.

[69] A. C. Yao. How to Generate and Exchange Secrets. In *IEEE Symposium on Foundations of Computer Science*, pages 162-167, 1986.

[70] M. Yung and Y. Zhao. Interactive Zero-Knowledge with Restricted Random Oracles. In *S. Halevi and T. Rabin (Ed.): Theory of Cryptography (TCC) 2006, LNCS 3876*, pages 21-40, Springer-Verlag, 2006.

[71] M. Yung and Y. Zhao. Generic and Practical Resettable Zero-Knowledge in the Bare Public-Key Model. In *M. Naor (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2007, LNCS 4515*, pages 116-134. Springer-Verlag, 2007.

[72] Y. Zhao, J. B. Nielsen, R. Deng and D. Feng. Generic yet Practical ZK Arguments from any Public-Coin HVZK. Electronic Colloquium on Computational Complexity, 12(162), 2005.

[73] A. C. Yao, M. Yung and Y. Zhao. Concurrent Knowledge-Extraction in the Public-Key Model. Electronic Colloquium on Computational Complexity (ECCC), Report No. 2007/002.

A Related Works

The concept of non-malleability is introduced by Dolve, Dwork and Naor in the seminal work of [28]. The work of [28] also presents non-constant-round non-malleable commitment and zero-knowledge protocols. CNMZK with a poly-logarithmic round complexity is achieved in the plain model [6]. Constant-round non-malleable coin-tossing protocol in the plain model (and accordingly, constant-round non-malleable zero-knowledge arguments for $\mathcal{NP}$ and commitment schemes by combining the result of [21]) is achieved by Barak [2]. The non-malleable coin-tossing protocol of [2] employs non-black-box techniques (introduced in [1]) in a critical way. Parallel coin-tossing , which can be viewed as a restricted version of concurrent coin-tossing, were studied in [52, 50].

A large number of concurrent non-malleable (and the strongest, universal composable) cryptographic protocols are developed in the common reference/random string model, where a common reference/random string is selected trustily by a trusted third party and is known to all players (e.g., [25, 37, 26, 21, 14, 19], etc). In particular, concurrent non-malleability for any functionality can be implemented in the common random string (CRS) model [21, 14].

The situation for general adaptive CNM feasibility in the bare public-key model turns out to be quite subtle and somewhat confused. There are several works that deal with the specific CNMZK protocols in the BPK model [64, 22, 65]. But, as clarified in this work (in Appendix D), the CNM formulations in existing works are incomplete or flawed. Also, no previous protocols in the BPK model or the plain model are known to be CNM secure against CMIM with full adaptive input selection. Actually, the possibility of CNM with adaptive input selection in the BPK model itself turns out to be a subtle issue, and was not clarified in existing works.

B The CMIM Setting in the BPK Model for Interactive Argument Systems

In this section, we give a detailed description of the CMIM setting in the BPK model, in accordance with interactive argument systems with players of fixed roles. The formalization can be directly extended to any two-party protocol with interchangeable roles.

The bare public-key (BPK) model. Let $\mathcal{R}^P_{KEY}$ be an $\mathcal{NP}$-relation validating the public-key and secret-key pair (PK_P, SK_P) generated by honest provers, i.e., $\mathcal{R}^P_{KEY}(PK_P, SK_P) = 1$ indicates that SK_P is a valid secret-key of PK_P. Similarly, let $\mathcal{R}^V_{KEY}$ be an $\mathcal{NP}$-relation validating the public-key and secret-key pair (PK_V, SK_V) generated by honest verifiers, i.e., $\mathcal{R}^V_{KEY}(PK_V, SK_V) = 1$ indicates that SK_V is a valid secret-key of PK_V. In the following formalization, we assume each honest player is of fixed player role.

Then, an interactive protocol $\langle P, V\rangle$ for an $\mathcal{NP}$-language $\mathcal{L}$ in the BPK model w.r.t. key-validating relations $\mathcal{R}^P_{KEY}$ and $\mathcal{R}^V_{KEY}$, consists of the following:

1. The interactions between P and V can be divided into two stages. The first stage is called *key-generation stage* in which each player registers a public-key in a public file F; at the end of the key-generation stage, the *proof stage* starts, where any pair of prover and verifier can interact. All algorithms have access to the same public file F output by the key-generation stage.

2. On security parameter 1^n, the *public file* F, structured as a collection of $poly(n)$ records, for a polynomial $poly(\cdot)$: $\{(id_1, PK_{id_1}), (id_2, PK_{id_2}), \cdots (id_{poly(n)}, PK_{id_{poly(n)}})\}$. F is empty at the beginning and is updated by players during the key-generation stage. As we assume players be of fixed roles, for presentation simplicity, we also denote $F = \{PK_I^{(1)}, PK_I^{(2)}, \cdots, PK_I^{(poly(n))}\}$, such that for any i, $1 \leq i \leq poly(n)$, $PK_I^{(i)}$ denotes a prover-key if $I = P$ or a verifier-key if $I = V$. The *same* version of the public file F obtained at the end of the key-generation stage will be used during the proof stage. That is, the public file F to be used in proof stages remains intact with that output at the end of key-generation stage.

3. An honest *prover* P is a pair of deterministic polynomial-time algorithm (P_1, P_2), where P_1 operates in the key-generation stage and P_2 operates in the proof stage. On input a security parameter 1^n and a random tape r_{P_1}, P_1 generates a key pair (PK_P, SK_P) satisfying $\mathcal{R}^P_{KEY}(PK_P, SK_P) = 1$, registers PK_P in the public file F as its public-key while keeping the corresponding secret key SK_P in secret. Denote by $\mathcal{K}_P$ the set of all legitimate (in accordance with $\mathcal{R}^P_{KEY}$) public-keys generated by $P_1(1^n)$, that is, $\mathcal{K}_P$ contains all possible legitimate prover public-key generated on security parameter n. Then, in the proof stage, on inputs (PK_P, SK_P), and $poly(n)$-bit string $x \in \mathcal{L}$, an auxiliary input w, a public file F and a verifier public-key $PK_V^{(j)} \in F$, and a random tape r_P, P_2 performs an interactive protocol with the verifier of $PK_V^{(j)}$ in the proof stage.

4. An honest *verifier* V is a pair of deterministic polynomial-time algorithm (V_1, V_2), where V_1 operates in the key-generation stage and V_2 operates in the proof stage. On input a security parameter 1^n and a random tape r_{V_1}, V_1 generates a key pair (PK_V, SK_V) satisfying $\mathcal{R}^V_{KEY}(PK_V, SK_V) = 1$, registers PK_V in the public file F as its public-key while keeping the corresponding secret key SK_V in secret. Denote by $\mathcal{K}_V$ the set of all legitimate (in accordance with $\mathcal{R}^V_{KEY}$) public-keys generated by $V_1(1^n)$, that is, $\mathcal{K}_V$ contains all possible legitimate verifier public-key generated on security parameter n. On inputs (PK_V, SK_V), the public file F and a prover public-key $PK_P^{(j)} \in F$, and a $poly(n)$-bit x and a random tape r_{V_2}, V_2 performs the interactive protocol with (the proof stage of) the prover of $PK_P^{(j)}$, and outputs "accept x" or "reject x" at the end of this protocol. *We stress that as the role of the honest verifier with its public-key is not interchangeable in the BPK model, the honest verifier with its public-key may prove the knowledge of its secret-key, but will never prove anything else.*

Notes: We remark that, though each player is allowed to register public-keys in the public-file in the original formulation of the BPK model [12], for some cryptographic tasks, e.g., concurrent and resettable zero-knowledge, only requiring verifiers to register public-keys suffices. In these cases provers' keys may not be used, or $\mathcal{K}_P$ can be just empty. Our formulation of the BPK model is for the general case, and provers' registered public-keys play an essential role for achieving CNM security with *full* adaptive input selection (to be addressed later). Also note that in the above formulation, honest players are of fixed roles. For protocols with players of interchangeable roles, the direct extension approach is to let each player register a pair of public-keys (PK_P, PK_V) and explicitly indicate its role in protocol executions.

The CMIM adversary. The CMIM adversary $\mathcal{A}$ in the BPK model is a probabilistic polynomial-time (PPT) algorithm that can act both as a prover and as a verifier, both in the key-generation stage and in the main proof stage.

In the key-generation stage, on 1^n and some auxiliary input $z \in \{0,1\}^*$ and a pair of honestly generated public-keys (PK_P, PK_V) generated by the honest prover and verifier, $\mathcal{A}$ outputs a set of public-keys, denoted by F', together with some auxiliary information τ to be used in the proof-stage (in particular τ can include z and a priori information about the secret-keys of honest players (SK_P, SK_V)). Then the public file F used in proof state is set to be $F' \cup \{PK_P, PK_V\}$. That is, $\mathcal{A}$ has complete control of the public file F. Here, we remark that, in general, the input to $\mathcal{A}$ in order to generate F' could be a set of public-keys generated by many honest provers and verifiers, rather than a single pair of public-keys (PK_P, PK_V) generated by a single honest prover and a single honest verifier. The formulation with a unique pair of honestly generated public-keys is only for presentation simplicity.

In the proof stage, on inputs (F, τ) $\mathcal{A}$ can concurrently interact with any polynomial number of instances of the honest prover of public-key PK_P in *left interaction part*. The interactions with each instance of the honest prover of PK_P is called a *left session*, in which $\mathcal{A}$ plays the role of verifier with a public-key $PK_V^{(j)} \in F$; *Simultaneously*, $\mathcal{A}$ interacts with any polynomial number of instances of the honest verifier PK_V in *right interaction part*. The interactions with each instance of the honest verifier of PK_V is called a *right session*, where it plays the role of prover with a public-key $PK_P^{(j)} \in F$. Here, all honest prover and verifier instances are working independently, and answer messages sent by $\mathcal{A}$ promptly.

Specifically, polynomially many concurrent sessions of the proof stage of the same protocol $\langle P, V\rangle$ take place in an asynchronous setting (say, over the Internet), and all the *unauthenticated* communication channels (among all the concurrently executing instances of $\langle P, V\rangle$) are controlled by the PPT adversary $\mathcal{A}$. This means that the honest prover instances cannot directly communicate with the honest verifier instances in the proof stages, since all communication messages are done through the adversary. The adversary $\mathcal{A}$, controlling the scheduling of messages in both parts of CMIM, can decide to simply relay the messages between any prover instance in the left part and the corresponding verifier instance in the right part. But, it can also decide to block, delay, divert, or change messages arbitrarily at its wish. For CMIM-adversary with adaptive input selection, $\mathcal{A}$ can further set the inputs to left sessions adaptively based on its view (besides adaptively setting inputs to right sessions). A CMIM adversary is called $s(n)$-CMIM adversary, for a positive polynomial $s(\cdot)$, if the adversary involves at most $s(n)$ concurrent sessions in each part of the CMIM setting and registers at most $s(n)$ public-keys in F', where n is the security parameter.

For presentation simplicity and without loss of generality, we have made the following conventions:

- We assume all honest prover instances are of the same public-key PK_P and all honest verifier instances are of the same public-key PK_V. That is, $\mathcal{A}$ concurrently interacts on the left with honest prover instances of the same public-key PK_P and on the right with honest verifier instances of the same public-key PK_V. And, the file F' generated by $\mathcal{A}$ is only based on $\{PK_P, PK_V\}$.
- The session number in left interaction part is equal to the session number in right interaction part, i.e., both of them are $s(n)$.

We remark that both the security model and the security analysis in this work can be easily extended to the general case: multiple different honest prover and verifier instances with multiple different public-keys, and different session numbers in left interactions and right interactions. We prefer the simplified formulation for the reason that it much simplifies the presentation and security analysis.

More formally, with respect to a protocol $\langle P, V\rangle$ for an $\mathcal{NP}$-language $\mathcal{L}$ with $\mathcal{NP}$-relation $\mathcal{R_L}$, an $s(n)$-CMIM adversary $\mathcal{A}$'s attack in the BPK model is executed in accordance with the following experiment $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, X, W, z)$, where $X = \{x_1, \cdots, x_{s(n)}\}$ and $W = \{w_1, \cdots, w_{s(n)}\}$ are vectors of $s(n)$ elements such that $x_i \in L \cap \{0,1\}^{poly(n)}$ and $(x_i, w_i) \in \mathcal{R_L}$, $1 \leq i \leq s(n)$:

$$\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, X, W, z)$$

Honest prover-key generation. $(PK_P, SK_P) \longleftarrow P_1(1^n)$.

Honest verifier-key generation. $(PK_V, SK_V) \longleftarrow V_1(1^n)$.

Preprocessing stage of the CMIM. $\mathcal{A}$, on inputs 1^n, auxiliary input $z \in \{0,1\}^*$ and honest player keys (PK_P, PK_V), outputs (F', τ), where F' is a list of, at most $s(n)$, public-keys and τ is some auxiliary information to be transferred to the proof stage of $\mathcal{A}$. Then, the public file to be used in the proof stage is: $F = F' \cup \{PK_P, PK_V\}$.

Proof stage of the CMIM. $\mathcal{A}$ continues its execution by taking (F, τ) as additional input, and may start (at most) $s(n)$ sessions in either the left CMIM interaction part or the right CMIM interaction part. At any time during this stage, $\mathcal{A}$ can do one of the following four actions.

- Deliver to V a message for an already started right session.
- Deliver to P a message for an already started left session.
- Start a new i-th left session, $1 \le i \le s(n)$: $\mathcal{A}$ indicates a key $PK_V^{(j)} \in F$ to the honest prover P (of public-key PK_P). The honest prover P then initiates a new session with (the predetermined) input (x_i, w_i) and the verifier of $PK_V^{(j)}$ (pretended by $\mathcal{A}$).

  For CMIM-adversary with (traditional) adaptive input selection, besides $PK_V^{(j)}$ the CMIM adversary $\mathcal{A}$ indicates to P, adaptively based on its view, a statement $\tilde{x}_i \in \{0,1\}^{poly(n)}$ as the input of as the i-th left session. In this case, we require that the membership of $\tilde{x}_i \in \mathcal{L} \cup \{0,1\}^{poly(n)}$ can be efficiently checked, otherwise, the experiment may render an $\mathcal{NP}$-membership oracle to $\mathcal{A}$. In case $\tilde{x}_i \in \mathcal{L} \cup \{0,1\}^{poly(n)}$ (that can be efficiently checked), then a witness $\tilde{w}_i$ such that $(\tilde{x}^l, \tilde{w}^l) \in \mathcal{R_L}$ is given to the prover instance of P; Then, on input (x_i, w_i) the honest prover P interacts with the verifier of $PK_V^{(j)}$ (pretended by $\mathcal{A}$).
- Start a new i-th right session: the CMIM adversary $\mathcal{A}$ chooses, adaptively based on its view from the CMIM attack, a $poly(n)$-bit string $\hat{x}_i$, and indicates a key $PK_P^{(j)} \in F$ to the honest verifier V of public-key PK_V; Then, the honest verifier V then initiates a new session, interacts with the prover of public-key $PK_P^{(j)}$ (pretended by $\mathcal{A}$) on input $(1^n, \hat{x}_i)$ in which $\mathcal{A}$ is trying to convince of the (possibly false) statement "$\hat{x}_i \in L$".
- Output a special "end attack" symbol within time polynomial in n.

We denote by $view_{\mathcal{A}}(1^n, X, z)$ the random variable describing the view of $\mathcal{A}$ in this experiment $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, X, W, z)$, which includes its random tape, the (predetermined) input vector X, the auxiliary string z, all messages it receives including the public-keys (PK_P, PK_V) and all messages sent by honest prover and verifier instances in the proof stages. For any $(PK_P, SK_P) \in \mathcal{R}^P_{KEY}$ and $(PK_V, SK_V) \in \mathcal{R}^V_{KEY}$, we denote by $view_{\mathcal{A}}^{P(SK_P),V(SK_V)}(1^n, X, z, PK_P, PK_V)$ the random variable describing the view of $\mathcal{A}$ specific to (PK_P, PK_V), which includes its random tape, the auxiliary string z, the (specific) (PK_P, PK_V), and all messages it receives from the instances of $P(1^n, SK_P)$ and $V(1^n, SK_V)$ in the proof stages.

Note that in all cases, the honest prover and verifier instances answer messages from $\mathcal{A}$ promptly. We stress that in different left or right sessions the honest prover and verifier instances use independent random-tapes in the proof stages. *The adversary's goal is to complete a right session with statement different from that of any left session, for which the verifier accepts even if the adversary actually does not know a witness for the statement being proved.*

C Basic Definitions and Tools

Basic notation. We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. If S is a finite set then $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. By $[R_1; \cdots; R_n : v]$ we denote the set of values of v that a random variable can assume, due to the distribution determined by the sequence of random processes $R_1, R_2, \cdots, R_n$. By $\Pr[R_1; \cdots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$.

Let $\langle P, V\rangle$ be a probabilistic interactive protocol, then the notation $(y_1, y_2) \leftarrow \langle P(x_1), V(x_2)\rangle(x)$ denotes the random process of running interactive protocol $\langle P, V\rangle$ on common input x, where P has private input x_1, V has private input x_2, y_1 is P's output and y_2 is V's output. We assume w.l.o.g. that the output of both parties P and V at the end of an execution of the protocol $\langle P, V\rangle$ contains a transcript of the communication exchanged between P and V during such execution.

The security of cryptographic primitives and tools, presented throughout this work, is defined with respect to uniform polynomial-time algorithms (equivalently, polynomial-size circuits). When it comes to non-uniform security, we refer to non-uniform polynomial-time algorithms (equivalently, families of polynomial-size circuits).

On a security parameter n (also written as 1^n), a function $\mu(\cdot)$ is negligible if for every polynomial $p(\cdot)$, there exists a value N such that for all $n > N$ it holds that $\mu(n) < 1/p(n)$. Let $X = \{X(n, z)\}_{n\in N, z\in\{0,1\}^*}$ and $Y = \{Y(n, z)\}_{n\in N, z\in\{0,1\}^*}$ be distribution ensembles. Then we say that X and Y are computationally (resp., statistically) indistinguishable, if for every probabilistic polynomial-time (resp., any, even power-unbounded) algorithm D, for all sufficiently large n's, and every $z \in \{0, 1\}^*$, $|\Pr[D(n, z, X(n, z)) = 1] - \Pr[D(n, z, Y(n, z)) = 1]|$ is negligible in n.

Definition C.1 (one-way function) *A function $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ is called a one-way function (OWF) if the following conditions hold:*

1. *Easy to compute: There exists a (deterministic) polynomial-time algorithm A such that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*
2. *Hard to invert: For every probabilistic polynomial-time PPT algorithm A', every positive polynomial $p(\cdot)$, and all sufficiently large n's, it holds $\Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$, where U_n denotes a random variable uniformly distributed over $\{0, 1\}^n$.*

Definition C.2 (interactive argument/proof system) *A pair of interactive machines, $\langle P, V\rangle$, is called an interactive argument system for a language $\mathcal{L}$ if both are probabilistic polynomial-time (PPT) machines and the following conditions hold:*

- *Completeness. For every $x \in \mathcal{L}$, there exists a string w such that for every string z, $\Pr[\langle P(w), V(z)\rangle(x) = 1] = 1$.*
- *Soundness. For every polynomial-time interactive machine P^*, and for all sufficiently large n's and every $x \notin \mathcal{L}$ of length n and every w and z, $\Pr[\langle P^*(w), V(z)\rangle(x) = 1]$ is negligible in n.*

An interactive protocol is called a proof *for $\mathcal{L}$, if the soundness condition holds against any (even power-unbounded) P^* (rather than only PPT P^*). An interactive system is called a public-coin system if at each round the prescribed verifier can only toss coins and send their outcome to the prover.*

Definition C.3 (witness indistinguishability WI [35]) *Let $\langle P, V\rangle$ be an interactive system for a language $\mathcal{L} \in \mathcal{NP}$, and let $\mathcal{R}_{\mathcal{L}}$ be the fixed $\mathcal{NP}$ witness relation for $\mathcal{L}$. That is, $x \in \mathcal{L}$ if there exists a w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$. We denote by $view^{P(w)}_{V^*(z)}(x)$ a random variable describing the transcript of all messages exchanged between a (possibly malicious) PPT verifier V^* and the honest prover P in an execution of the protocol on common input x, when P has auxiliary input w and V^* has auxiliary input z. We say that $\langle P, V\rangle$ is witness indistinguishable for $\mathcal{R}_{\mathcal{L}}$ if for every PPT interactive machine V^*, and every two sequences $W^1 = \{w^1_x\}_{x\in L}$ and $W^2 = \{w^2_x\}_{x\in L}$ for sufficiently long x, so that $(x, w^1_x) \in \mathcal{R}_{\mathcal{L}}$ and $(x, w^2_x) \in \mathcal{R}_{\mathcal{L}}$, the following two probability distributions are computationally indistinguishable by any non-uniform polynomial-time algorithm: $\{x, view^{P(w^1_x)}_{V^*(z)}(x)\}_{x\in\mathcal{L}, z\in\{0,1\}^*}$ and*

$\{x, view_{V^*(z)}^{P(w_x^2)}(x)\}_{x\in\mathcal{L},\, z\in\{0,1\}^*}$. *Namely, for every non-uniform polynomial-time distinguishing algorithm D, every polynomial $p(\cdot)$, all sufficiently long $x \in \mathcal{L}$, and all $z \in \{0,1\}^*$, it holds that*

$$|\Pr[D(x, z, view_{V^*(z)}^{P(w_x^1)}(x) = 1] - \Pr[D(x, z, view_{V^*(z)}^{P(w_x^2)}(x) = 1]| < \frac{1}{p(|x|)}$$

It is interesting to note that the WI property preserves against adaptive concurrent composition [35, 34, 36, 21].

Definition C.4 (strong witness indistinguishability SWI [38]) *Let $\langle P, V\rangle$ and all other notations be as in Definition C.3. We say that $\langle P, V\rangle$ is* strongly witness-indistinguishable *for $\mathcal{R}_\mathcal{L}$ if for every PPT interactive machine V^* and for every two probability ensembles $\{X_n^1, Y_n^1, Z_n^1\}_{n\in N}$ and $\{X_n^2, Y_n^2, Z_n^2\}_{n\in N}$, such that each $\{X_n^i, Y_n^i, Z_n^i\}_{n\in N}$ ranges over $(\mathcal{R}_\mathcal{L} \times \{0,1\}^*) \cap (\{0,1\}^n \times \{0,1\}^* \times \{0,1\}^*)$, the following holds: If $\{X_n^1, Z_n^1\}_{n\in N}$ and $\{X_n^2, Z_n^2\}_{n\in N}$ are computationally indistinguishable, then so are $\{\langle P(Y_n^1), V^*(Z_n^1)\rangle(X_n^1)\}_{n\in N}$ and $\{\langle P(Y_n^2), V^*(Z_n^2)\rangle(X_n^2)\}_{n\in N}$.*

WI vs. SWI: It is clarified in [39] that the notion of SWI actually refers to issues that are fundamentally different from WI. Specifically, the issue is whether the interaction with the prover helps V^* to distinguish some auxiliary information (which is indistinguishable without such an interaction). Significantly different from WI, SWI does *not* preserve under concurrent composition. More details about SWI are referred to [39]. An interesting observation, as clarified later, is: the protocol composing commitments and SWI can be itself regular WI. Also note that any zero-knowledge protocol is itself SWI [39].

Definition C.5 (zero-knowledge ZK [44, 38]) *Let $\langle P, V\rangle$ be an interactive system for a language $\mathcal{L} \in \mathcal{NP}$, and let $\mathcal{R}_\mathcal{L}$ be the fixed $\mathcal{NP}$ witness relation for $\mathcal{L}$. That is, $x \in \mathcal{L}$ if there exists a w such that $(x, w) \in \mathcal{R}_\mathcal{L}$. We denote by $view_{V^*(z)}^{P(w)}(x)$ a random variable describing the contents of the random tape of V^* and the messages V^* receives from P during an execution of the protocol on common input x, when P has auxiliary input w and V^* has auxiliary input z. Then we say that $\langle P, V\rangle$ is zero-knowledge if for every probabilistic polynomial-time interactive machine V^* there exists a probabilistic (expected) polynomial-time oracle machine S, such that for all sufficiently long $x \in \mathcal{L}$ the ensembles $\{view_{V^*}^{P(w)}(x)\}_{x\in\mathcal{L}}$ and $\{S^{V^*}(x)\}_{x\in\mathcal{L}}$ are computationally indistinguishable. Machine S is called a ZK simulator for $\langle P, V\rangle$. The protocol is called statistical ZK if the above two ensembles are statistically close (i.e., the variation distance is eventually smaller than $\frac{1}{p(|x|)}$ for any positive polynomial p). The protocol is called perfect ZK if the above two ensembles are actually identical (i.e., except for negligible probabilities, the two ensembles are equal).*

Definition C.6 (system for argument/proof of knowledge [38, 8]) *Let $\mathcal{R}$ be a binary relation and $\kappa : N \to [0, 1]$. We say that a probabilistic polynomial-time (PPT) interactive machine V is a knowledge verifier for the relation $\mathcal{R}$ with knowledge error κ if the following two conditions hold:*

- *Non-triviality: There exists an interactive machine P such that for every $(x, w) \in \mathcal{R}$ all possible interactions of V with P on common input x and auxiliary input w are accepting.*
- *Validity (with error κ): There exists a polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every interactive machine P^*, every $x \in \mathcal{L}_\mathcal{R}$, and every $w, r \in \{0,1\}^*$, machine K satisfies the following condition:*

  Denote by $p(x, w, r)$ the probability that the interactive machine V accepts, on input x, when interacting with the prover specified by $P^*_{x,w,r}$ (where $P^*_{x,w,r}$ denotes the strategy of P^* on common input x, auxiliary input w and random-tape r). If $p(x, w, r) > \kappa(|x|)$, then, on input x and with oracle access to $P^*_{x,w,r}$, machine K outputs a solution $w' \in \mathcal{R}(x)$ within an expected number of steps bounded by

  $$\frac{q(|x|)}{p(x, w, r) - \kappa(|x|)}$$

  The oracle machine K is called a knowledge extractor.

An interactive argument/proof system $\langle P, V\rangle$ such that V is a knowledge verifier for a relation $\mathcal{R}$ and P is a machine satisfying the non-triviality condition (with respect to V and $\mathcal{R}$) is called a system for argument/proof of knowledge (AOK/POK) for the relation $\mathcal{R}$.

The above definition of POK is with respect to *deterministic* prover strategy. POK also can be defined with respect to *probabilistic* prover strategy. It is recently shown that the two definitions are equivalent for all natural cases (e.g., POK for $\mathcal{NP}$-relations) [8].

Definition C.7 (pseudorandom functions PRF) *On a security parameter n, let $d(\cdot)$ and $r(\cdot)$ be two positive polynomials in n. We say that*

$$\{f_s : \{0,1\}^{d(n)} \longrightarrow \{0,1\}^{r(n)}\}_{s\in\{0,1\}^n}$$

is a pseudorandom function ensemble if the following two conditions hold:

1. *Efficient evaluation: There exists a polynomial-time algorithm that on input s and $x \in \{0,1\}^{d(|s|)}$ returns $f_s(x)$.*
2. *Pseudorandomness: For every probabilistic polynomial-time oracle machine A, every polynomial $p(\cdot)$, and all sufficiently large n's, it holds:*

$$|\Pr[A^{F_n}(1^n) = 1] - \Pr[A^{H_n}(1^n) = 1]| < \frac{1}{p(n)}$$

   where F_n is a random variable uniformly distributed over the multi-set $\{f_s\}_{s\in\{0,1\}^n}$, and H_n is uniformly distributed among all functions mapping $d(n)$-bit-long strings to $r(n)$-bit-long strings.

PRFs can be constructed under any one-way function [40, 38]. The current most practical PRFs are the Naor-Reingold implementations under the factoring (Blum integers) or the decisional Diffie-Hellman hardness assumptions [62]. The computational complexity of computing the value of the Naor-Reingold functions at a given point is about two modular exponentiations and can be further reduced to only two multiple products modulo a prime (without any exponentiations!) with natural preprocessing, which is great for practices involving PRFs.

Definition C.8 (statistically/perfectly binding bit commitment scheme) *A pair of PPT interactive machines, $\langle P, V\rangle$, is called a perfectly binding bit commitment scheme, if it satisfies the following:*

Completeness. *For any security parameter n, and any bit $b \in \{0,1\}$, it holds that* $\Pr[(\alpha, \beta) \leftarrow \langle P(b), V\rangle(1^n); (t,(t,v)) \leftarrow \langle P(\alpha), V(\beta)\rangle(1^n) : v = b] = 1$.

Computationally hiding. *For all sufficiently large n's, any PPT adversary V^*, the following two probability distributions are computationally indistinguishable:* $[(\alpha, \beta) \leftarrow \langle P(0), V^*\rangle(1^n) : \beta]$ *and* $[(\alpha', \beta') \leftarrow \langle P(1), V^*\rangle(1^n) : \beta']$.

Perfectly Binding. *For all sufficiently large n's, and* any *adversary P^*, the following probability is negligible (or equals 0 for perfectly-binding commitments):* $\Pr[(\alpha, \beta) \leftarrow \langle P^*, V\rangle(1^n); (t,(t,v)) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n); (t',(t',v')) \leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n) : v, v' \in \{0,1\} \bigwedge v \neq v']$.

*That is, no (*even computational power unbounded*) adversary P^* can decommit the same transcript of the commitment stage both to 0 and 1.*

Below, we recall some classic perfectly-binding commitment schemes.

One-round perfectly-binding (computationally-hiding) commitments can be based on any one-way permutation OWP [9, 42]. Loosely speaking, given a OWP f with a hard-core predict b (cf. [38]), on a security parameter n one commits a bit σ by uniformly selecting $x \in \{0,1\}^n$ and sending $(f(x), b(x) \oplus \sigma)$ as a commitment, while keeping x as the decommitment information.

Statistically-binding commitments can be based on any one-way function (OWF) but run in two rounds [60, 48]. On a security parameter n, let $PRG : \{0,1\}^n \longrightarrow \{0,1\}^{3n}$ be a pseudorandom generator, the Naor's OWF-based two-round public-coin perfectly-binding commitment scheme works as follows: In the first round, the commitment receiver sends a random string $R \in \{0,1\}^{3n}$ to the committer. In the second round, the committer uniformly selects a string $s \in \{0,1\}^n$ at first; then to commit a bit 0 the committer sends $PRG(s)$ as the commitment; to commit a bit 1 the committer sends $PRG(s) \oplus R$ as the commitment.

One-round perfectly-binding (computationally-hiding) commitments can be based on any one-way permutation OWP. Loosely speaking, given a OWP f with a hard-core predict b, on a security parameter n one commits a bit σ by uniformly selecting $x \in \{0,1\}^n$ and sending $(f(x), b(x) \oplus \sigma)$ as a commitment, while keeping x as the decommitment information.

For practical perfectly-binding commitment scheme, in this work we use the DDH-based ElGamal non-interactive commitment scheme [31]. To commit to a value $v \in Z_q$, the committer randomly selects $u, r \in Z_q$, computes $h = g^u \mod p$ and sends $(h, \bar{g} = g^r, \bar{h} = g^v h^r)$ as the commitment. The decommitment information is (r, v). Upon receiving the commitment $(h, \bar{g}, \bar{h})$, the receiver checks that $h, \bar{g}, \bar{h}$ are elements of order q in Z_p^*. It is easy to see that the commitment scheme is of perfectly-binding. The computational hiding property is from the DDH assumption on the subgroup of order q of Z_p^*. We also note that Micciancio and Petrank presented another implementation of DDH-based perfectly-binding commitment scheme with advanced security properties [57].

A commitment scheme C is called *linear*, if for sufficiently large n and any string $x \in \{0,1\}^n$ both $|C(x, s)|$ (i.e., the length of the commitment to x using random coins s) and $|s|$ (i.e., the length of s) are bounded by $O(n)$.

Commit-then-SWI: Consider the following protocol composing a statistically-binding commitment and SWI:

Common input: $x \in \mathcal{L}$ for an $\mathcal{NP}$-language $\mathcal{L}$ with corresponding $\mathcal{NP}$-relation $\mathcal{R}_{\mathcal{L}}$.

Prover auxiliary input: w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$.

The protocol: consisting of two stages:

- **Stage-1:** The prover P computes and sends $c_w = C(w, r_w)$, where C is a statistically-binding commitment and r_w is the randomness used for commitment.
- **Stage-2:** Define a new language $\mathcal{L}' = \{(x, c_w) | \exists (w, r_w)\ s.t.\ c_w = C(w, r_w) \wedge \mathcal{R}_{\mathcal{L}}(x, w) = 1\}$. Then, P proves to V that it knows a witness to $(x, c_w) \in \mathcal{L}'$, by running a SWI protocol for $\mathcal{NP}$.

One interesting observation for the above commit-then-SWI protocol is that commit-then-SWI is itself a regular WI for $\mathcal{L}$.

Proposition C.1 *Commit-then-SWI is itself a regular WI for the language $\mathcal{L}$.*

Proof (of Proposition C.1). For any PPT malicious verifier V^*, possessing some auxiliary input $z \in \{0,1\}^*$, and for any $x \in \mathcal{L}$ and two (possibly different) witnesses (w_0, w_1) such that $(x, w_b) \in \mathcal{R}_{\mathcal{L}}$ for both $b \in \{0,1\}$, consider the executions of commit-then-SWI: $\langle P(w_0), V^*(z)\rangle(x)$ and $\langle P(w_1), V^*(z)\rangle(x)$.

Note that for $\langle P(w_b), V^*(z)\rangle(x)$, $b \in \{0,1\}$, the input to SWI of Stage-2 is $(x, c_{w_b} = C(w_b, r_{w_b}))$, and the auxiliary input to V^* at the beginning of Stage-2 is (x, c_{w_b}, z). Note that (x, c_{w_0}, z) is indistinguishable from (x, c_{w_1}, z). Then, the regular WI property of the whole composed protocol is followed from the SWI property of Stage-2. □

C.1 Adaptive tag-based one-left-many-right non-malleable statistical zero-knowledge argument of knowledge (SZKAOK)

Let $\{\langle P_{TAG}, V_{TAG}\rangle(1^n)\}_{n \in N, TAG \in \{0,1\}^{O(n)}}$ be a family of argument systems for an $\mathcal{NP}$-language $\mathcal{L}$ specified by $\mathcal{NP}$-relation $\mathcal{R}_{\mathcal{L}}$. For each security parameter n and $TAG \in \{0,1\}^{O(n)}$, $\langle P_{TAG}, V_{TAG}\rangle(1^n)$ is an instance of the protocol $\langle P, V\rangle$, which is indexed by TAG and works for inputs in $\mathcal{L} \cup \{0,1\}^{p(n)}$, where $p(\cdot)$ is some polynomial.

We consider an experiment EXPE$(1^n, x, TAG, z)$, where 1^n is the security parameter, $x \in \mathcal{L} \cup \{0,1\}^{p(n)}$, $TAG \in \{0,1\}^{O(n)}$ and $z \in \{0,1\}^*$. (The input (x, TAG) captures the predetermined input and tag of the prover instance in the following left MIM part, and the string $z \in \{0,1\}^*$ captures the auxiliary input to the following MIM adversary $\mathcal{A}$.) In the experiment EXPE$(1^n, x, TAG, z)$, on input $(1^n, x, TAG, z)$, an adaptive input-selecting one-left-many-right MIM adversary $\mathcal{A}$ is simultaneously participating in two interaction parts:

The left MIM part: in which $\mathcal{A}$ chooses $(\tilde{x}^l, \widetilde{TAG}^l)$ based on its view from both the left session and all right sessions, satisfying that: the membership of $\tilde{x}^l \in \mathcal{L} \cup \{0,1\}^{p(n)}$ can be efficiently checked [1] and $\widetilde{TAG}^l \in \{0,1\}^{O(n)}$; In case $\tilde{x}^l \in \mathcal{L} \cup \{0,1\}^{p(n)}$ (that can be efficiently checked), then a witness $\tilde{w}^l$ such that $(\tilde{x}^l, \tilde{w}^l) \in \mathcal{R}_\mathcal{L}$ is given to the prover instance $P_{\widetilde{TAG}^l}$, and $\mathcal{A}$ interacts, playing the role of the verifier $V_{\widetilde{TAG}^l}$, with the prover instance $P_{\widetilde{TAG}^l}(\tilde{x}, \tilde{w})$ on common input $\tilde{x}^l$. The interactions with $P_{\widetilde{TAG}^l}(\tilde{x}^l, \tilde{w}^l)$ is called the left session. Note that, $\mathcal{A}$ can just set $(\tilde{x}^l, \widetilde{TAG}^l)$ to be (x, TAG), which captures the case of predetermined input and tag to left session.

The right CMIM part: in which $\mathcal{A}$ *concurrently* interacts with $s(n)$, for a polynomial $s(\cdot)$, verifier instances: $V_{\widetilde{TAG}_1^r}(\tilde{x}_1^r)$, $V_{\widetilde{TAG}_2^r}(\tilde{x}_2^r)$, $\cdots$, $V_{\widetilde{TAG}_{s(n)}^r}(\tilde{x}_{s(n)}^r)$, where $(\widetilde{TAG}_i^r, \tilde{x}_i^r)$, $1 \leq i \leq s(n)$, are set by $\mathcal{A}$ (at the beginning of each session) adaptively based on its view (in both the left session and all the right sessions) satisfying $\tilde{x}_i^r \in \{0,1\}^{p(n)}$ and $\widetilde{TAG}_i^r \in \{0,1\}^{O(n)}$. The interactions with the instance $V_{\widetilde{TAG}_i^r}(\tilde{x}_i^r)$ is called the i-th right session, in which $\mathcal{A}$ plays the role of $P_{\widetilde{TAG}_i^r}$.

Denote by $view_\mathcal{A}(1^n, x, TAG, z)$ the random variable describing the view of $\mathcal{A}$ in the above experiment EXPE$(1^n, x, TAG, z)$, which includes the input $(1^n, x, TAG, z)$, its random tape, and all messages received in the one left session and the $s(n)$ right sessions.

Then, we say that the family of argument systems $\{\langle P_{TAG}, V_{TAG}\rangle(1^n)\}_{n\in N, TAG\in\{0,1\}^{O(n)}}$ is adaptive tag-based one-left-many-right non-malleable SZKAOK with respect to tags of length $O(n)$, if for any PPT adaptive input-selecting one-left-many-right MIM adversary $\mathcal{A}$ defined above, there exists an expected polynomial-time algorithm S, such that for any sufficiently large n, any $x \in \mathcal{L} \cup \{0,1\}^{p(n)}$ and $TAG \in \{0,1\}^{O(n)}$, and any $z \in \{0,1\}^*$, the output of $S(1^n, x, TAG, z)$ consists of two parts (str, sta) such that the following hold, where we denote by $S_1(1^n, x, TAG, z)$ (the distribution of) its first output str.

- **Statistical simulatability.** The following ensembles are statistically indistinguishable: $\{view_\mathcal{A}(1^n, x, TAG, z)\}_{n\in N, x\in\mathcal{L}\cup\{0,1\}^{p(n)}, TAG\in\{0,1\}^{O(n)}, z\in\{0,1\}^*}$ and $\{S_1(1^n, x, TAG, z)\}_{n\in N, x\in\mathcal{L}\cup\{0,1\}^{p(n)}, TAG\in\{0,1\}^{O(n)}, z\in\{0,1\}^*}$
- **Knowledge extraction.** sta consists of a set of $s(n)$ strings, $\{w_1, w_2, \cdots, w_{s(n)}\}$, satisfying the following:
  - For any i, $1 \leq i \leq s(n)$, if the i-th right session in str is aborted or with a tag identical to that of the left session, then $w_i = \perp$;
  - Otherwise, i.e., the i-th right session in str is successful with $\widetilde{TAG}_i^r \neq \widetilde{TAG}^l$, then $(\tilde{x}_i^r, w_i) \in \mathcal{R}_\mathcal{L}$, where $\tilde{x}_i^r$ is the input to the i-th right session in str.

Pass-Rosen ZK (PRZK). The PRZK (with some specified length parameter $l(n) \geq O(n^3)$) developed in [67, 68] is the only known *constant-round* adaptive tag-based one-left-many-right non-malleable SZKAOK. Furthermore, PRZK is *public-coin* and can be of *perfect* ZK. We note that in

[1] We remark, for our purpose of security analysis in Section 5.1, it is necessary, as well as sufficient, to require the membership of the statement $\tilde{x}^l$ chosen by $\mathcal{A}$ can be efficiently checked; otherwise, the experiment may render an $\mathcal{NP}$-membership oracle to $\mathcal{A}$.

[67, 68], the tag length is just specified to be the security parameter n (in this case, the length parameter is specified as $l(n) \geq 2n^3 + n$), but a closer investigation shows that the PRZK can be extended to work for tags of length $O(n)$ and inputs of length $poly(n)$. The works of [67, 68] do not explicitly consider adaptive input and tag selection for the one left-session, but a closer investigation shows that the security analysis presented in [67, 68] works also for this more general case.

D Formulating CNMZK in the Public-Key Model, Revisited

Traditional CNMZK formulation roughly is the following: for any PPT CMIM adversary $\mathcal{A}$ of traditional input selecting capability (as clarified in Appendix B), there exists a PPT simulator/extractor S such that S outputs the following: (1) A simulated transcript that is indistinguishable from the real view of the CMIM adversary in its CMIM attacks. (2) For a successful right session on a common input $\hat{x}$ different from those of left sessions, S can output a corresponding $\mathcal{NP}$-witness of $\hat{x}$.

The requirement (1) intuitively captures that any advantage of $\mathcal{A}$ can get from concurrent left and right interactions can also be got by S itself alone without any interactions, i.e, $\mathcal{A}$ gets no extra advantage by the CMIM attacks. The requirement (2) intuitively captures that for any different statement that $\mathcal{A}$ convinces of V in one of right sessions, $\mathcal{A}$ must "know" a witness.

The formulations of CNM in the public-key model in existing works ([64, 22, 65]) essentially directly bring the above traditional CNM formulation into the public-key setting, but with the following difference: S will simulate the key-generation phases of all honest verifiers. Put in other words, in its simulation/extration S actually takes the corresponding secret-keys of honest verifiers.

We start clarifying the subtleties of CNM in the public-key model by showing a CMIM attack on the CNMZK in the BPK model proposed in [22]. The CMIM attack allows the CMIM adversary to successfully convince the honest verifier of some $\mathcal{NP}$ statements but without knowing any witness to the statement being proved.

D.1 CMIM attacks on the CNMZK proposed in [22]

Let us first recall the protocol structure of the protocol of [22].

Key-generation. Let (KG_0, Sig_0, Ver_0) and (KG_1, Sig_1, Ver_1) be two signature schemes that secure against adaptive chosen message attacks. On a security parameter 1^n, each verifier V randomly generates two pair $(verk_0, sigk_0)$ and $(verk_1, sigk_1)$ by running KG_0 and KG_1 respectively, where $verk$ is the signature verification key and $sigk$ is the signing key. V publishes $(verk_0, verk_1)$ as its public-key while keeping $sigk_b$ as its secret-key for a randomly chosen b from $\{0, 1\}$ (V discards $sigk_{1-b}$). The prover does not possess public-key.

Common input. An element $x \in \mathcal{L}$ of length $poly(n)$, where $\mathcal{L}$ is an $\mathcal{NP}$-language that admits Σ-protocols.

The main-body of the protocol. The main-body of the protocol consists of the following three phases:

Phase-1. The verifier V proves to P that it knows either $sigk_0$ or $sigk_1$, by executing the (partial witness-independent) Σ_{OR}-protocol [15] on $(verk_0, verk_1)$ in which V plays the role of knowledge prover. Denote by a_V, e_V, z_V, the first-round, the second-round and the third-round message of the Σ_{OR}-protocol of this phase respectively. Here e_V is the random challenge sent by the prover to the verifier.

If V successfully finishes the Σ_{OR}-protocol of this phase and P accepts, then goto Phase-2. Otherwise, P aborts.

Phase-2. P generates a key pair (sk, vk) for a one-time strong signature scheme. Let COM be a commitment scheme. The prover randomly selects random strings $s, r \in \{0, 1\}^{poly(n)}$, and computes $C = COM(s, r)$ (that is, P commits to s using randomness r). Finally, P sends (C, vk) to the verifier V.

Phase-3. By running a Σ_{OR}-protocol, P proves to V that it knows either a witness w for $x \in \mathcal{L}$ OR the value committed in C is a signature on the message of vk under either $verk_0$ or $verk_1$. Denote by a_P, e_P, z_P, the first-round, the second-round and the third-round message of the Σ_{OR} of Phase-3. Finally, P computes a one-time strong signature δ on the whole transcript with the signing key sk generated in Phase-2.

Verifier's decision. V accepts if and only if the Σ_{OR}-protocol of Phase-3 is accepting, and δ is a valid signature on the whole transcript under vk.

Note: The actual implementation of the DDL protocol combines rounds of the above protocol. But, it is easy to see that round-combination does not invalidate the following attacks.

D.1.1 The CMIM attack

We show a special CMIM attack in which the adversary $\mathcal{A}$ only participate the right concurrent interactions with honest verifiers (i.e., there are no concurrent left interactions in which $\mathcal{A}$ concurrently interacts with honest provers).

The following CMIM attack enables $\mathcal{A}$ to malleate the interactions of Phase-1 of one session into a successful conversation of another concurrent session for different (but verifier's public-key related) statements without knowing any corresponding $\mathcal{NP}$-witnesses.

Let $\hat{L}$ be any $\mathcal{NP}$-language admitting a Σ-protocol that is denoted by $\Sigma_{\hat{L}}$ (*in particular, $\hat{L}$ can be an empty set*). For an honest verifier V with its public-key $PK = (verk_0, verk_1)$, we define a new language $\mathcal{L} = \{(\hat{x}, verk_0, verk_1) | \exists w \ s.t. \ (\hat{x}, w) \in \mathcal{R}_{\hat{L}} \text{ OR } w = sigk_b \text{ for } b \in \{0,1\}\}$. Note that for any string $\hat{x}$ (whether $\hat{x} \in \hat{L}$ or not), the statement "$(\hat{x}, verk_0, verk_1) \in \mathcal{L}$" is always true as $PK = (verk_0, verk_1)$ is honestly generated. Also note that $\mathcal{L}$ is a language that admits Σ-protocols (as Σ_{OR}-protocol is itself a Σ-protocol). Now, we describe the concurrent interleaving and malleating attack, in which $\mathcal{A}$ successfully convinces the honest verifier of the statement "$(\hat{x}, verk_0, verk_1) \in \mathcal{L}$" for *any arbitrary $poly(n)$-bit string $\hat{x}$* (*even when $\hat{x} \notin \hat{L}$*) by concurrently interacting with V (with public-key $(verk_0, verk_1)$) in two sessions as follows.

1. $\mathcal{A}$ initiates the first session with V. After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the first session on common input $(verk_0, verk_1)$ (i.e., V's public-key), $\mathcal{A}$ suspends the first session.

2. $\mathcal{A}$ initiates a second session with V, and works just as the honest prover does in Phase-1 and Phase-2 of the second session. We denote by C, vk the Phase-2 message of the second session, where C is the commitment to a random string and vk is the verification key of the one-time strong signature scheme generated by $\mathcal{A}$ (*note that $\mathcal{A}$ knows the corresponding signing key sk as (vk, sk) is generated by itself*). When $\mathcal{A}$ moves into Phase-3 of the second session and needs to send V the first-round message, denoted by a_P, of the Σ_{OR}-protocol of Phase-3 of the second session *on common input $(\hat{x}, verk_0, verk_1)$*, $\mathcal{A}$ does the following:

   - $\mathcal{A}$ first runs the SHVZK simulator of $\Sigma_{\hat{L}}$ (i.e., the Σ-protocol for $\hat{L}$) [18] on $\hat{x}$ to get a simulated conversation, denoted by $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$, for the (*possibly false*) statement "$\hat{x} \in \hat{L}$".
   - $\mathcal{A}$ runs the SHVZK simulator of the Σ-protocol for showing that the value committed in C is a signature on vk under one of $(verk_0, verk_1)$ to get a simulated conversation, denoted by (a_C, e_C, z_C).
   - $\mathcal{A}$ sets $a_P = (a_{\hat{x}}, a'_V, a_C)$ and sends a_P to V as the first-round message of the Σ_{OR}-protocol of Phase-3 of the second session, where a'_V is the one received by $\mathcal{A}$ in the first session.
   - After receiving the second-round message of Phase-3 of the second session, i.e., the random challenge e_P from V, $\mathcal{A}$ suspends the second session.

3. $\mathcal{A}$ continues the first session, and sends $e'_V = e_P \oplus e_{\hat{x}} \oplus e_C$ as the second-round message of the Σ_{OR}-protocol of Phase-1 of the first session.

4. After receiving the third-round message of the Σ_{OR}-protocol of Phase-1 of the first session, denoted by z'_V, $\mathcal{A}$ suspends the first session again.

5. $\mathcal{A}$ continues the execution of the second session again, sends to $z_P = ((e_{\hat{x}}, z_{\hat{x}}), (e'_V, z'_V), (e_C, z_C))$ to V as the third-round message of the Σ_{OR}-protocol of the second session.

6. Finally, $\mathcal{A}$ applies sk on the whole transcript *of the second session* to get a (one-time strong) signature δ, and sends δ to V

Note that $(a_{\hat{x}}, e_{\hat{x}}, z_{\hat{x}})$ is an accepting conversation for the (possibly false) statement "$\hat{x} \in \hat{L}$", (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of either $sigk_0$ or $sigk_1$, (a_C, e_C, z_C) is an accepting conversation for showing that the value committed in C is a signature on vk under one of $(verk_0, verk_1)$. Furthermore, $e_{\hat{x}} \oplus e'_V \oplus e_C = e_P$, and δ is a valid (one-time strong) signature on the transcript of the second session.This means that, from the viewpoint of V, $\mathcal{A}$ successfully convinced V of the statement "$(\hat{x}, verk_0, verk_1) \in \mathcal{L}$" in the second session *but without knowing any corresponding $\mathcal{NP}$-witness*!

D.2 Reformulating CNMZK in the BPK model

In light of the above CMIM attacks, we highlight a key difference between the CMIM setting in the public-key model and the CMIM setting in the standard model.

THE KEY DIFFERENCE: For CMIM setting in the standard model, honest verifiers are PPT algorithms. In this case, normal CNM formulation only considers the extra advantages the CMIM adversary can get from concurrent left sessions, as the actions of honest verifiers in right sessions can be efficiently emulated perfectly; But, for CMIM setting in the public-key model, the honest verifier possesses secret value (i.e, its secret-key) that can *not* be computed out efficiently from the public-key. In other words, in this case an CMIM adversary can get extra advantages both from the left sessions and *from the right sessions*. This is a crucial difference between CMIM settings for standard model and public-key model, which normal formulation of CNM does not capture. The CMIM attack on the protocol of [22] clearly demonstrates this difference.

With the above key difference in mind, we investigate reformulating the CNM notion in the public-key model. Above all, besides requiring the ability of simulation/extraction, we need to mandate that for any CMIM-adversary the witnesses extracted for right sessions are "*independent*" of the secret-key used by the simulator/extractor S (who emulates honest verifiers in the simulation/extraction). Such property is named concurrent non-malleable knowledge-extraction independence (CNMKEI). CNMKEI is formulated by extending the formulation of concurrent knowledge-extraction (CKE) of [73] into the more complicated CMIM setting (the CKE notion is formulated with adversaries only interacting with honest verifiers but without interacting with provers). Roughly, the CNMKEI is formulated as follows.

CNMKEI IN THE PUBLIC-KEY MODEL: We require that for any PPT CMIM-adversary $\mathcal{A}$ in the BPK model, there exists a PPT simulator/extractor S such that the following holds: $\Pr[\mathcal{R}(\widehat{W}, SK_V, str)] = 1$ is negligibly close to $\Pr[\mathcal{R}(\widehat{W}, SK'_V, str)] = 1$ for any polynomial-time computable relation $\mathcal{R}$, where SK'_V is some element *randomly and independently* distributed over the space of SK_V, str is the simulated transcript indistinguishable from the real view of $\mathcal{A}$, and $\widehat{W}$ are the joint witnesses extracted to successful right sessions in str. Here, for some right session that is aborted (due to CMIM adversary abortion or verifier verification failure) or is of common input identical to that of one left session, the corresponding witness to that right session is set to be a special symbol $\perp$.

The formal formulation of the reformulated CNMZK definition in the BPK model is presented below:

Definition D.1 (CNMZK in the public-key model) *We say that a protocol $\langle P, V\rangle$ is* concurrently non-malleable zero-knowledge *in the BPK model w.r.t. a class of admissible languages $\mathcal{L}$ and some key-validating relations $\mathcal{R}^P_{KEY}$ and $\mathcal{R}^V_{KEY}$, if for any positive polynomial $s(\cdot)$, any s-CMIM adversary $\mathcal{A}$*

defined in Appendix B, there exist a pair of (expected) polynomial-time algorithms $S = (S_{KEY}, S_{PROOF})$ (the simulator) and E (the extractor) such that for any sufficiently large n, any auxiliary input $z \in \{0,1\}^*$, any $\mathcal{NP}$-relation $\mathcal{R}_L$ (indicating an admissible language $L \in \mathcal{L}$), and any polynomial-time computable relation $\mathcal{R}$ (with components drawn from $\{0,1\}^* \cup \{\perp\}$), the following hold, in accordance with the experiment $\mathsf{Expt}_{\mathbf{CNM}}(1^n, X, z)$ described below (page 29):

$\mathsf{Expt}_{\mathbf{CNM}}(1^n, X, z)$

Honest prover key-generation:
$(PK_P, SK_P) \longleftarrow P_1(1^n)$. Denote by $\mathcal{K}_L$ the set of all legitimate public-keys generated by $P_1(1^n)$. *Note that the execution of P_1 is independent from the simulation below. In particular, only the public-key PK_P is passed on to the simulator.*

The simulator $S = (S_{KEY}, S_{PROOF})$:
$(PK_V, SK_V, SK'_V) \longleftarrow S_{KEY}(1^n)$, where the distribution of (PK_V, SK_V) is identical with that of the output of the key-generation stage of the honest verifier V_1, $\mathcal{R}^V_{KEY}(PK_V, SK_V) = \mathcal{R}^V_{KEY}(PK_V, SK'_V) = 1$ and the distributions of SK_V and SK'_V are identical and *independent*. In other words, SK_V and SK'_V are two random and independent secret-keys corresponding to PK_V.

$(str, sta) \longleftarrow S_{PROOF}^{\mathcal{A}(1^n,\ X,\ PK_P,\ PK_V,\ z)}(1^n, X, PK_P, PK_V, SK_V, z)$. That is, on inputs $(1^n, X, PK_P, PK_V, SK_V, z)$ and with oracle access to $\mathcal{A}(1^n, X, PK_P, PK_V, z)$ (defined in accordance with the experiment $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, X, W, z)$ described in Appendix B), the simulator S outputs a simulated transcript str, and some state information sta to be transformed to the knowledge-extractor E. *Note that S does not know the secret-key SK_P of honest prover, that is, S can emulate the honest prover only from its public-key PK_P.*

For any $X \in L^{s(n)}$ and $z \in \{0,1\}^*$, we denote by $S_1(1^n, X, z)$ the random variable str (in accordance with above processes of P_1, S_{KEY} and S_{PROOF}). For any $X \in L^{s(n)}$, $PK_P \in \mathcal{K}_P$ and $(PK_V, SK_V) \in \mathcal{R}^V_{KEY}$ and any $z \in \{0,1\}^*$, we denote by $S_1(1^n, X, PK_P, PK_V, SK_V, z)$ the random variable describing the first output of $S_{PROOF}^{\mathcal{A}(1^n,\ X,\ PK_P,\ PK_V,\ z)}(1^n, X, PK_P, PK_V, SK_V, z)$ (i.e., str specific to (PK_P, PK_V, SK_V)).

The knowledge-extractor E:
$\widehat{W} \longleftarrow E(1^n, sta, str)$. On (sta, str), E outputs a list of witnesses to (different right) statements whose validations are successfully conveyed in right sessions in str, where each of these statements is different from the statements of left sessions.

- **Simulatability.** *The following ensembles are indistinguishable:* $\{S_1(1^n, X, PK_P, PK_V, SK_V, z)\}_{X \in L^{s(n)}, PK_P \in \mathcal{K}_P, (PK_V, SK_V) \in \mathcal{R}^V_{KEY}, z \in \{0,1\}^*}$ *and* $\{view_{\mathcal{A}}^{P(SK_P), V(SK_V)}(1^n, X, PK_P, PK_V, z)\}_{X \in L^{s(n)}, PK_P \in \mathcal{K}_P, (PK, SK) \in R_{KEY}, z \in \{0,1\}^*}$ *(defined in accordance with the experiment $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, X, W, z)$ described in Appendix B). This in particular implies the probability ensembles $\{S_1(1^n, X, z)\}_{X \in L^{s(n)}, z \in \{0,1\}^*}$ and $\{view_{\mathcal{A}}(1^n, X, z)\}_{X \in L^{s(n)}, z \in \{0,1\}^*}$ are indistinguishable.*

- **Secret-key independent knowledge-extraction.** *E, on inputs $(1^n, str, sta)$, outputs witnesses to all (different right) statements successfully proved in accepting right sessions in str (with each of the statements different from those of left sessions). Specifically, E outputs a list of strings $\widehat{W} = (\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_{s(n)})$, satisfying the following:*

– $\hat{w}_i$ *is set to be* $\perp$, *if the* i*-th right session in* str *is not accepting (due to abortion or verifier verification failure) or the common input of the* i*-th right session is identical with that of one of left sessions, where* $1 \leq i \leq s(n)$.

– Correct knowledge-extraction for (individual) statements: *In any other cases, with overwhelming probability* $(\hat{x}_i, \hat{w}_i) \in \mathcal{R}_{\mathcal{L}}$, *where* $\hat{x}_i$ *is the statement selected by* P^* *for the* i*-th right session in* str *and* $\mathcal{R}_{\mathcal{L}}$ *is the* $\mathcal{NP}$*-relation for the admissible language* $L \in \mathcal{L}$ *set by* P^* *for right sessions in* str.

– concurrent non-malleable knowledge extraction independence (CNMKEI): $\Pr[\mathcal{R}(SK_V, \widehat{W}, str) = 1]$ *is negligibly close to* $\Pr[\mathcal{R}(SK'_V, \widehat{W}, str) = 1]$. *This in particular implies that the distributions of* (PK_V, SK_V, str) *and* (PK_V, SK'_V, str) *are indistinguishable (by considering* PK_V *encoded in* $\widehat{W}$*).*

The probabilities are taken over the randomness of P_1, *the randomness of* S *in the key-generation stage (i.e., the randomness for generating* (PK_V, SK_V, SK'_V)*) and in all proof stages, the randomness of* E, *and the randomness of* $\mathcal{A}$.

Note that the above CNM formulation in the public-key model implies both concurrent ZK for concurrent prover security in the public-key model (note that S emulates the honest prover without knowing its secret-key), and concurrent knowledge-extraction for concurrent verifier security in the public-key model formulated in [73]. The CNM formulation follows the simulation-extraction approach of [68], and extends the CKE formulation of [73] into the more complex CMIM setting. We remark that, as clarified, mandating the CNMKEI property is crucial for correctly formulating CNM security in the public-key model. We also note that the above CNMZK definition in the BPK model can be trivially extended to a tag-based formalization version

D.3 Discussions and clarifications

Existing CNM formulations in the public-key model do not capture CNMKEI. The CNM formulation in the work [64] uses the indistinguishability-based approach of [68]. Specifically, in the CNM formulation of [64], two experiments are defined (page 19 of [64]): a real experiment w.r.t. a real public-key of an honest verifier (here, denoted PK_V), in which a CMIM adversary mounts CMIM attacks; a simulated experiment run by a simulator/extractor S w.r.t. a simulated public-key (here, denoted PK_S), in which S accesses $\mathcal{A}$ and takes a simulated secret-key SK_S. The CNM is then formulated as follows: the distribution of all witnesses used by $\mathcal{A}$ in right sessions in the real experiment is indistinguishable from the distribution of the witnesses used by $\mathcal{A}$ in right sessions in the simulated experiment. *Note that [64] does not require the simulator/extractor to output a simulated indistinguishable transcript.* That is, the CNM formulation of [64] does *not* automatically imply concurrent zero-knowledge.

It appears that the CNM formulation of [64] has already dealt with the issue of knowledge-extraction independence. But, a careful investigation shows that it does not. The reason is as follows:

Firstly, in the real experiment the statements selected by the CMIM adversary $\mathcal{A}$ for both left and right sessions can be maliciously related to PK_V (e.g., some function of PK_V), and thus the witnesses extracted for right sessions of the real experiment could be potentially dependent on the secret-key SK_V used by honest players. Note that, as witnessed by the above concurrent interleaving and malleating attack on the CNMZK protocol of [22], when extracted witnesses are maliciously dependent on SK_V knowledge-extraction does not necessarily capture the intuition that $\mathcal{A}$ does "know" the witnesses extracted. Similarly, as in the simulated experiment S uses SK_S in simulation/extraction, the witness extracted in the simulated experiment could also be maliciously dependent on SK_S. That is, both the witnesses extracted in real experiment and in the simulated experiment may be maliciously dependent on SK_V and SK_S respectively, *but the distributions of them still can be indistinguishable as the distributions of* SK_V *and* SK_S *are identical!*

The CNMZK formulations in the subsequent works of [22, 65]formulation following the simulation/extraction approach, which is incomplete for correctly capture CNM security in the public-key model as clarified above.

CNM with full adaptive input selection. The above CNMZK formulation does not explicitly specify the input-selecting capabilities of the CMIM adversary. According to the clarifications presented in Section 3, there are four kinds of CNM security to be considered: CNM security against CMIM with predetermined inputs, CNM security against CMIM with adaptive input selection, CNM security against CMIM with predetermined left-session inputs but full adaptive input selection on the right, and CNM security against CMIM with full adaptive input selection.

We briefly note that no previous protocols in the BPK model were proved to be CNM-secure against even CMIM with predetermined left-session inputs but full adaptive input selection on the right (i.e., the inputs to left sessions are predetermined and the CMIM adversary only sets inputs to right sessions in the fully adaptive way), needless to say to be CNM secure against CMIM with full adaptive input selection. Specifically, the standard simulation-extraction paradigm for showing CNM security fails, in general, when the CMIM adversary is allowed the capability of full adaptive input selection.

In more detail, the standard simulation-extraction paradigm for establishing CNM security works as follows: the simulator first outputs an indistinguishable simulated transcript; and then extracts the witnesses to (different) inputs of successful right sessions appearing in the simulated transcript, *one by one sequentially*, by applying some assured underlying knowledge-extractor. This paradigm can work for CMIM adversary with the capability of traditional adaptive input selection, as the input to each right session is fixed at the beginning of the right session; Thus, applying knowledge-extractor on the right session does not change the statement of the session, which has appeared and is fixed in the simulated transcript.

But, for CMIM adversary of fully adaptive input selection, the standard simulation-extraction paradigm fails in general in this case. In particular, considering the adversary always sets inputs to right sessions only at the last message of each right session, such case applies to both of the two illustrative natural protocol examples presented in Section 3: composing coin-tossing and NIZK, and the Feige-Shamir-ZK-like protocols. In this case, when we apply knowledge-extractor on a successful right session, the statement of this session will however also be changed, which means that the extractor may never extract witness to the same statement appearing and being fixed in the simulated transcript.

On the possibility of CNMZK with adaptive input selection in the BPK model. The possibility of CNMZK with adaptive (not necessarily to be fully adaptive) input selection in the BPK model turns also out to be a quite subtle issue. In particular, we note that (traditional) adaptive input selection was highlighted for the CNMZK in [64], but the updated version of [65] are w.r.t. predetermined prover inputs (such subtleties were not clarified in [64, 65]. It appears that, as noted recently in [66], the existence of CNMZK with adaptive (needless to say fully adaptive) input selection in the BPK model might potentially violate Lindell's impossibility results on concurrent composition with adaptive input selection [56, 54]. This raised the question that: whether constant-round CNMZK protocols (particularly in accordance with our CNMZK formulation) with adaptive input selection exists in the BPK model (or, whether it is possible at least)?

A careful investigation shows that constant-round CNMZK with adaptive input selection could still be possible in the BPK model, and actually our work does imply such protocols with the strongest *full* adaptive input selection. Below, we give detailed clarifications in view of Lindell's impossibility results of [56, 54]. Lindell's impossibility results of [56, 54] hold for concurrent (self or general) composition of protocols securely realizing (large classes of) functionalities enabling (bilateral) bit transmission. The Zero-Knowledge functionality $((x,w),\lambda) \mapsto (\lambda,(x,R(x,w)))$ enables unilateral bit transformation from prover to verifier. But, when a CNMZK protocol *in the plain model* is considered, where the CMIM adversary can play both the role of prover and the role of the verifier (note that the honest verifier can be perfectly emulated by the CMIM adversary in the plain model), it actually amounts to realize an extended version of ZK functionality *with interchangeable roles* that does enable bilateral bit transformation in this case. This implies that CNMZK with adaptive input selection is impossible in the plain model.

The ZK (not necessarily CNMZK) protocol for an $\mathcal{NP}$-language $\mathcal{L}$ in the BPK model essentially amounts to securely realizing the following functionality: $((x,w),(PK_V,SK_V)) \mapsto ((PK_V,\mathcal{R}^V_{KEY}(PK_V, SK_V)),(x,\mathcal{R}_L(x,w)))$ that enables bilateral bit transmission. This means that when adaptive input selection is allowed both for prover inputs and *verifier's keys*, which implies the verifier's keys and thus the public file output by the key-generation stage are not fixed but are set accordingly by the CMIM adversary in order to transmit bits from honest verifiers to honest provers, even concurrent ZK (needless to say CNMZK) may not exist in the BPK model! We highlight some key points that still could allow the possibilities of CNMZK with adaptive input selection in the BPK model:

- **Disabling bit transformation from honest verifiers to other players:** Note that: in key-generation stage, the keys of *honest* verifiers are generated independently by the honest verifiers themselves and cannot be set adaptively by the CMIM adversary; In the proof stages, the keys of honest verifiers (actually all keys in the public file) cannot be modified by the CMIM adversary, as we assume the public file used in the proof stages remains the same output at the end of key-generation stage; Furthermore, in the BPK setting we assume the role of honest verifiers with honestly generated keys is fixed. That is, honest verifiers may prove the knowledge of their corresponding secret-keys, but they never prove anything else.

  Putting all together, it means that honest verifiers instantiated with their public-keys cannot be impersonated and emulated by the CMIM adversary, and their inputs (i.e., the keys generated in key-generation stage and then fixed and remaining unchanged for proof stages) and their prescribed actions and player role in the proof stages are not influenced by the CMIM adversary. This disables bit transmission from honest verifiers to other players, which implies that the existence of CNMZK with adaptive input selection in the BPK model could still not violate Lindell's impossibility results.

- **Disabling bit transformation from other players to honest provers:** For a protocol in the BPK model, the public-keys registered by honest provers and the public-keys registered by honest verifiers can be of different types, and the use of honest-prover keys and the use of honest-verifier keys in protocol implementation can also be totally different. Such differences can be on the purpose of protocol design, as demonstrated with our CNMCT implementation. Then, for honest provers of fixed role in the BPK model, though the CMIM adversary can enable, by adaptive input selection, bit transmissions from honest provers to other players, but, in the BPK model, the CMIM adversary may not enable bit transmissions from other players to honest provers.

- **Concurrent self composition vs. concurrent general composition in the BPK model:** We further consider a more general case for any two-party protocol $\langle P,V\rangle$ in the BPK model. Suppose there are some players of fixed role, and some players of interchangeable roles (i.e., players who can serve both as prover and as verifier). The direct way for a player in the BPK model to be of interchangeable roles is to register a pair of keys (PK_P,PK_V) and to explicitly indicate its role, i.e., prover or verifier, in the run of each session. Then, according to the analysis of [56, 54], the run of any arbitrary external protocol executed among players of interchangeable roles can be emulated, by a CMIM adversary capable of adaptive input selection, in the setting of concurrent self composition of the protocol $\langle P,V\rangle$ among those players. But, the external protocol executions involving honest players of fixed roles, however, are not necessarily be able to be emulated by self-composition of the protocol involving the honest players of fixed roles. This implies that, as long as there are honest players of fixed roles in the BPK model, concurrent self-composition with adaptive input selection in the BPK system does not necessarily imply concurrent general composability.

A tradeoff. The above clarifications also pose a tradeoff between players' roles and their CNM security levels in the BPK model: For stronger CNM security of adaptive input selection, honest players in the BPK model need to be of fixed roles; Of course, honest players can also choose to be of interchangeable roles for their own convenience, but with the caveat that CNM security against CMIM of adaptive input selection may lose (though CNM with predetermined inputs can still remain). In other words, whether to be of fixed role or interchangeable role can be at the discretion of each honest player in the BPK model. If one is interested with the stronger CNM security against CMIM of (full) adaptive input selection, it is necessary for it to be of fixed role. A typical scenario of this case is: this player is a server, who normally plays the same role and takes higher priority of stronger security over Internet; However, if one is interested in the convenience of interchangeable role, it can simply register a pair of keys (PK_P, PK_V) and explicitly indicate its role in the run of each session, but with the caveat that its CNM security against CMIM of adaptive input selection may lose.

E Proof of Theorem 5.1

The description of the simulator. On security parameter 1^n, for any positive polynomial $s(\cdot)$ and any PPT $s(n)$-CMIM adversary $\mathcal{A}$ in the BPK model with auxiliary information $z \in \{0,1\}^*$, the simulator $S = (S_{KEY}, S_{PROOF})$, with respect to the honest left-player key-registration algorithm L_{KEY} and a CRS simulating algorithm $\mathcal{M}_{CRS}$, is re-depicted in Figure 3 (page 34) in order to ease reference. In the description, the notation of m denotes a message sent by the simulator (emulating honest players), and $\tilde{m}$ denotes the arbitrary message sent by the CMIM-adversary $\mathcal{A}$.

Notes on the CNM simulation: For any i, $1 \leq i \leq s(n)$, if in the i-th left (resp., right) session of the simulation $\mathcal{A}$ does not act accordingly or fails to provide a valid proof, then S aborts that session, and sets the output just to be $S_L^{(i)}$ (resp., $S_R^{(i)}$) and the state information to be $\tau_L^{(i)}$ (resp., $\tau_R^{(i)}$).

Note that in the right-session simulation, when a successful right-session is w.r.t. a left-player key $PK_L^{(j)} = PK_L$ the simulator does not try to extract the secret-key of PK_L. In the following analysis, we show that in this case, with overwhelming probability, the tag of Stage-5 of this successful right session is identical to that of Stage-5 of a left-session. As the tag of Stage-5 of a session consists of the session output (i.e., the coin-tossing output), this implies that the session output of this right-session is identical to that of one of left-sessions. Moreover, we show that with overwhelming probability each left-session output can appear, as session output, in at most one successful right-session.

In the unlikely event that $\mathcal{A}$ finishes a right session and the Stage-1 of a left-session simultaneously, both of which are w.r.t. uncovered public-keys, extracting SK_R in left simulation part takes priority (in this case, SK_L extraction in right simulation part is ignored in the current simulation repetition).

During any (of the at most $s(n) + 1$) simulation repetition, if S does not encounter secret-key extraction and does not stop due to Case-R1 failure or Case-R2 failure, then S stops whenever $\mathcal{A}$ stops, and sets str to be F and the view of $\mathcal{A}$ in this simulation repetition and $sta = (sta_L, sta_R)$ to be the according state-information.

Analysis of the CNM simulation

In order to establish the CNM security of the coin-tossing protocol depicted in Figure 1, according to the CNMCT definition of Definition 5.1, we need to show the following properties of the CNM simulator S described in Figure 3:

- S works in expected polynomial-time.
- The simulatability property, i.e., the output of S is computationally indistinguishable from the view of $\mathcal{A}$ in real CMIM attack.
- The property of strategy-restricted and predefinable randomness.
- The secret-key independence property.

In the following, we analyze the above four properties of the CNM simulator S case by case.

- S works in expected polynomial-time

External honest left-player key-generation: Let $(PK_L, SK_L) \longleftarrow L_{KEY}(1^n)$, where $PK_L = c$ and $SK_L = (\sigma, s_\sigma)$ such that $\sigma \in \{0,1\}^n$ and $s_\sigma \in \{0,1\}^{t(n)}$ and $c = C(\sigma, s_\sigma)$. This captures the fact that S does not know SK_L and can emulate the honest left-player with the same public-key PK_L.

Public-key file generation:
$S_{KEY}(1^n)$ perfectly emulates the key-generation stage of the honest right-player, getting $PK_R = (y_0 = f(s_0), y_1 = f(s_1))$ and $SK_R = s_b$ and $SK'_R = s_{1-b}$ for a random bit b. Then, S_{KEY} runs $\mathcal{A}(1^n, PK_L, PK_R, z)$ to get (F', τ), where F' is a set of at most $s(n)$ public-keys and τ is the state information to be used by the proof stage of $\mathcal{A}$. The public-key file to be used in the proof-stage is $F = F' \cup \{PK_L, PK_R\}$.

$\mathcal{S} \leftarrow \{(PK_R, SK_R)\}$ (i.e. initiate the set of covered keys $\mathcal{S}$ to be $\{(PK_R, SK_R)\}$).
On input $(1^n, F', PK_L, PK_R, SK_R, \tau)$ and running $\mathcal{A}(PK_L, PK_R, F', \tau)$ as a subroutine, the following process is run by S_{PROOF} repeatedly at most $s(n)+1$ times. In each simulation repetition, S uses fresh randomness and tries to either end with a successful simulation or cover a new public-key in $F - \mathcal{S}$.

Straight-line left simulation:
In the i-th left concurrent session (ordered by the time-step in which the first round of each session is played) between S and $\mathcal{A}$ in the left CMIM interaction part with respect to a public-key $PK_R^{(j)} = (y_0^{(j)}, y_1^{(j)}) \in \mathcal{K}_R$, $1 \le i, j \le s(n)$, S acts as follows:

In case $\mathcal{A}$ successfully finishes Stage-1 and $PK_R^{(j)} \in F' - \mathcal{S}$, the simulator ends the current repetition of simulation trial, and starts to extract a secret-key $SK_R^{(j)}$ such that $\mathcal{R}_{KEY}^R(PK_R^{(j)}, SK_R^{(j)}) = 1$, which is guaranteed by the AOK property of PRZK. Then, let $\mathcal{S} \leftarrow \mathcal{S} \cup \{(PK_R^{(j)}, SK_R^{(j)})\}$, and move to the next repetition with fresh randomness (but with the accumulated covered-key set $\mathcal{S}$ and the same public-key file F).

In case $\mathcal{A}$ successfully finishes Stage-1 and $PK_R^{(j)} \in \mathcal{S}$ (i.e., S has already learnt the secret-key $SK_R^{(j)}$), S randomly selects $r_l^{(i)\prime} \leftarrow \{0,1\}^n$ and sends $r_l^{(i)\prime}$ to $\mathcal{A}$ at Stage-2. After receiving Stage-3 message, denoted $\tilde{r}_r^{(i)}$, from $\mathcal{A}$, S invokes $\mathcal{M}_{CRS}(1^n)$ and gets the output denoted $(S_L^{(i)}, \tau_L^{(i)})$. S then sends $r^{(i)} = S_L^{(i)}$ as the Stage-4 message (rather than sending back $r^{(i)} = PRF_\sigma(r_l^{(i)\prime}) \oplus \tilde{r}_r^{(i)}$ as the honest left-player does), and sets $sta_L^{(i)} = \tau_L^{(i)}$. In Stage-5, S computes and sends $c_{crs}^{(i)} = C(SK_R^{(j)} || 0^{t(n)}, s_{crs}^{(i)})$ to $\mathcal{A}$ (rather than sending back $c_{crs}^{(i)} = C(\sigma || s_\sigma)$ as the honest left-player does), where $t(n)$ is the length of s_σ in SK_L. Finally, S finishes the PRZK of Stage-5 with $(SK_R^{(j)}, s_{crs}^{(i)})$ as its witness and $(PK_L, \tilde{r}_r^{(i)}, S_L^{(i)})$ as the tag.

Straight-line right simulation:
In the i-th right concurrent session (ordered by the time-step in which the first round of each session is played) between S and $\mathcal{A}$ in the right CMIM interaction part with respect to a public-key $PK_L^{(j)} = c^{(j)} \in \mathcal{K}_L$, $1 \le i, j \le s(n)$, S acts as follows:

S perfectly emulates honest right-player in Stage-1 of any right session, with SK_R as the witness to commit-then-PRZK and $(PK_L^{(j)}, PK_R)$ as the tag.

Case-R1: If $PK_L^{(j)} \in \mathcal{S}$ (i.e., S has already learnt the secret-key $SK_L^{(j)} = (\sigma^{(j)}, s_\sigma^{(j)})$), after receiving $\tilde{r}_l^{(i)\prime}$ from $\mathcal{A}$ at Stage-2, S runs $\mathcal{M}_{CRS}(1^n)$ and gets the output denoted $(S_R^{(i)}, \tau_R^{(i)})$, and then computes and sends $PRF_{\sigma^{(j)}}(\tilde{r}_l^{(i)\prime}) \oplus S_R^{(i)}$ as Stage-3 message, and goes further as the honest right-player does.

Case-R2: If $PK_L^{(j)} \notin \mathcal{S} \cup \{PK_L\}$, and $\mathcal{A}$ successfully finishes the i-th right session (in which S just perfectly emulates the honest right-player of PK_R), then the simulator S ends the current repetition of simulation trial, and starts to extract a secret-key $SK_L^{(j)}$ such that $\mathcal{R}_{KEY}^L(PK_L^{(j)}, SK_L^{(j)}) = 1$. *In case S fails to extract such $SK_L^{(j)}$, S stops the simulation, and outputs a special symbol $\perp$ indicating simulation failure. Such simulation failure is called Case-R2 failure.* In case S successfully extracts such $SK_L^{(j)}$, then let $\mathcal{S} \leftarrow \mathcal{S} \cup \{(PK_L^{(j)}, SK_L^{(j)})\}$, and move to the next repetition. If $PK_L^{(j)} = PK_L$, S just works as the honest right-player does.

Setting sta_R: For successful i-th right session, if the Stage-4 message $\tilde{r}^{(i)}$ is $S_R^{(i)}$ or $S_L^{(k)}$ for some k, $1 \le k \le s(n)$, then $sta_R^{(i)}$ is set accordingly to $\tau_R^{(i)}$ or $\tau_L^{(k)}$; otherwise, $sta_R^{(i)}$ is set to be $\perp$.

Figure 3: The CNM simulation

Note that S works for at most $s(n)+1$ repetitions. Then, pending on the ability of S to extract secret-key of uncovered public-keys in expected polynomial-time during each repetition (equivalently, within running-time inversely propositional to the probability of secret-key extraction event occurs), S will work in expected polynomial-time. The technique for covering public-keys follows that of [12, 5]. Below, we specify the secret-key extraction procedures in more details.

Right-player key coverage. Whenever S needs to extract the secret-key $SK_R^{(j)}$ corresponding to an uncovered public-key $PK_R^{(j)}$, due to successful Stage-1 of the i-th left session during the k-th simulation repetition w.r.t. covered key set $\mathcal{S}^{(k)}$, $1 \le i,j \le s(n)$ and $1 \le k \le s(n)+1$, we combine the CMIM adversary $\mathcal{A}$ and the simulation other than Stage-1 of the i-th left session (i.e., the public file F, the covered key set $\mathcal{S}^{(k)}$, the randomness $r_{\mathcal{A}}$ of $\mathcal{A}$, and the randomness $r_{\mathcal{S}}$ used by S except for that to be used in Stage-1 of the i-th left session) into an imaginary (deterministic) knowledge prover $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$. Note that, by the description of the CNM simulation depicted in Figure 3, the Stage-1 of the i-th left session is the *first successful* Stage-1 of a left session finished by $\mathcal{A}$ (during the k-th simulation repetition) with respect to an uncovered public-key not in $\mathcal{S}^{(k)}$. The knowledge-prover $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$ only interacts with a stand-alone knowledge-verifier of commit-then-PRZK, by running $\mathcal{A}$ internally and mimicking S with respect to $\mathcal{S}^{(k)}$ but with the following exceptions: (1) the messages belonging to the Stage-1 of the i-th left session are relayed between the internal $\mathcal{A}$ and the external stand-alone knowledge-verifier of PRZK; (2) $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$ ignores the events of secret-key extraction in right simulation part, i.e., successful right sessions with respect to uncovered (left-player) public-keys; (3) whenever $\mathcal{A}$ (run internally by $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$) successfully finishes, *for the first time*, Stage-1 of a left session w.r.t. an uncovered (right-player) public-key not in $\mathcal{S}^{(k)}$, $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$ just stops.

For any intermediate $\mathcal{S}^{(k)}$ used in the k-th simulation repetition, any $PK_R^{(j)} \notin \mathcal{S}^{(k)}$, any randomness $r_{\mathcal{A}}$ of $\mathcal{A}$ and any randomness $r_{\mathcal{S}}$ used by S except for that to be used in Stage-1 of the i-th left session, denote by p the probability (taken over the coins used by $\mathcal{S}$ for Stage-1 of the i-th left session) that the public-key used by $\mathcal{A}$ in Stage-1 of the i-th left session is $PK_R^{(j)}$, and furthermore, the Stage-1 of the i-th left session is the *first* successful execution of PRZK w.r.t. an uncovered public-key during the simulation of $\mathcal{S}$ w.r.t. covered-key set $\mathcal{S}^{(k)}$. That is, p is the probability, taken over the coins used by $\mathcal{S}$ for Stage-1 of the i-th left session (but for fixed other coins), of the event that $\mathcal{S}$ needs to cover $PK_R^{(j)} \notin \mathcal{S}^{(k)}$ in the i-th left session in its simulation w.r.t. $\mathcal{S}^{(k)}$. Clearly, with probability at least p, the knowledge prover $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$ successfully convinces the stand-alone knowledge verifier of $PK_R^{(j)}$. By the AOK property of PRZK and applying the knowledge-extractor on $\hat{P}^{(i,j)}_{(\mathcal{S}^{(k)},r_{\mathcal{A}},r_{\mathcal{S}})}$, the secret-key $SK_R^{(j)}$ will be extracted within running-time inversely propositional to p. Here, when p is negligible, standard technique, originally proposed in [41] and then deliberated in [52], has to be applied here (to estimate the value of p) in order to make sure expected polynomial-time knowledge-extraction. In more detail, the running-time of the naive approach to directly applying knowledge-extractor whenever such events occur is bounded by $T(n) = p \cdot \frac{q(n)}{p-\kappa(n)}$, where $\kappa(n)$ is the knowledge-error and $q(\cdot)$ is the polynomial related to the running time of the knowledge-extractor that is $\frac{q(n)}{p-\kappa(n)}$. The subtle point is: when p is negligible, $T(n)$ is not necessarily to be polynomial in n. The reader is referred to [41, 52] for the technical details of dealing with this issue.

Left-player key coverage.

The coverage procedure for uncovered (left-player) public-keys used by $\mathcal{A}$ in successful Stage-5 of right sessions can be described accordingly, similar to above right-player key coverage. The key point to note here is: for a successful right session with respect to an uncovered (left-player) public-key $PK_L^{(j)}$, the value extracted in expected polynomial-time is not necessarily to be the secret-key $SK_L^{(j)}$, though the value extracted must be either $SK_L^{(j)}$ or SK_R (i.e., the preimage of either y_0 or y_1) , where $PK_R = (y_0, y_1)$ is the simulated (right-player) public-key. That is, S may abort due to Case-R2 failure (*though it works in expected polynomial-time*). We show, in the following analysis of the simulatability property, Case-R2 failure occurs with at most negligible probability.

- Simulatability

For presentation simplicity, in the following analysis of simulatability we assume the first output of $\mathcal{M}_{CRS}$ is truly random string of length n, i.e., all $S_L^{(i)}$'s and $S_R^{(i)}$'s are truly random strings. The extension of the simulatability analysis to the case of pseudorandom output of $\mathcal{M}_{CRS}$ is direct.

Assuming truly random output of $\mathcal{M}_{CRS}$, there are three differences between the simulated transcript output by S and the view of $\mathcal{A}$ in real CMIM attack against the honest left-player of PK_L and the honest right-player of PK_R:

Truly random vs. pseudorandom Stage-4 messages: In simulation, the simulator S sends truly random string $r^{(i)} = S_L^{(i)}$ at Stage-4 of the i-th left session, for any i, $1 \leq i \leq s(n)$. But, the honest left-player sends a pseudorandom Stage-4 message, i.e., $r^{(i)} = PRF_\sigma(r_l^{(i)\prime}) \oplus \tilde{r}_r^{(i)}$, where $r_l^{(i)\prime}$ and $\tilde{r}_r^{(i)}$ are the Stage-2 and Stage-3 messages of the i-th left session.

Witness difference of Stage-5 of left sessions: For any i-th left session w.r.t. a public-key $PK_R^{(j)} \in \mathcal{S}$, the witness used by S in the commit-then-PRZK of Stage-5 is always the extracted secret-key $SK_R^{(j)}$, while the witness used by the honest left-player is always its secret-key SK_L.

Case-R2 failure: S may stop with simulation failure, due to invalid secret-key extraction in Case-R2 in the right simulation part.

We first show that, conditioned on Case-R2 failure does not occur, the output of S is indistinguishable from the real view of $\mathcal{A}$. Specifically, we have the following lemma:

Lemma E.1 *Conditioned on Case-R2 failure does not occur, the following ensembles are indistinguishable:* $\{S(1^n, z, PK_L, PK_R, SK_R)\}_{1^n, PK_L \in \mathcal{K}_L, (PK_R, SK_R) \in \mathcal{R}^R_{KEY}, z \in \{0,1\}^*}$ *(defined in Definition 5.1) and* $\{view_{\mathcal{A}}^{L(SK_L), R(SK_R)}(1^n, z, PK_L, PK_R)\}_{1^n, PK_L \in \mathcal{K}_L, (PK_R, SK_R) \in \mathcal{R}^R_{KEY}, z \in \{0,1\}^*}$ *(defined in accordance with the experiment* $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, z)$ *depicted in Appendix B, page 19).*

Proof (of Lemma E.1). We first note that, conditioned on Case-R2 failure does not occur and assuming the truly random output of $\mathcal{M}_{CRS}$, S perfectly emulates the honest right-player of PK_R in right simulation part.

The left two differences all are w.r.t. left session simulation. Intuitively, in real interaction the seed σ of PRF is committed into left-player public-key PK_L and is re-committed and proved concurrently in Stage-5 of left sessions, the CMIM adversary may potentially gain some knowledge about the random seed σ by concurrent interaction, which enabling it to set its Stage-3 messages of left sessions maliciously depending on the output of PRF_σ. Note that in real interaction, the Stage-4 messages sent by honest left-player are determined by the PRF seed and the Stage-2 messages. Thus, the Stage-4 messages of left sessions in real interaction may be distinguishable from truly random strings as sent by the simulator S in simulation. The still indistinguishability between the simulated transcript and the real view of $\mathcal{A}$ is proved by hybrid arguments.

We consider a hybrid mental experiment $\mathcal{H}$. $\mathcal{H}$ mimics $S(1^n, z, PK_L, PK_R, SK_R)$, with additionally possessing $SK_L = (\sigma, s_\sigma)$ and with the following exception: At Stage-4 of any left session, $\mathcal{H}$ just emulates the honest left-player by setting the Stage-4 message $r^{(i)}$ to be $PRF_\sigma(r_l^{(i)\prime}) \oplus \tilde{r}_r^{(i)}$ (rather than sending $S_L^{(i)}$ as S does); In Stage-5 of any left session w.r.t. a covered key $PK_R^{(j)}$ (for which $\mathcal{H}$ has already learnt the corresponding secret-key $SK_L^{(j)}$), $\mathcal{H}$ still emulates S by using the extracted secret-key $SK_R^{(j)}$ as the witness (specifically, it commits to $SK_R^{(j)}||0^t$ and finishes PRZK accordingly as the simulator S does).

The difference between the view of $\mathcal{A}$ in $\mathcal{H}$ and the view of $\mathcal{A}$ in the simulation of S lies in the difference of Stage-4 messages of left sessions. Suppose that the view of $\mathcal{A}$ in $\mathcal{H}$ is distinguishable from the view of $\mathcal{A}$ in the simulation of S, then it implies that there exists a PPT algorithm D that, given the commitment of the PRF seed, i.e., $PK_L = C(\sigma, s_\sigma)$, can distinguish the output of PRF_σ from truly random strings. Specifically, on input PK_L, D emulates $\mathcal{H}$ or S by having oracle access to PRF_σ or a truly random function; Whenever it needs to send Stage-4 message in a left session, it just queries its oracle with the Stage-2 message. Clearly, if the oracle is PRF_σ, then D perfectly emulates $\mathcal{H}$, otherwise (i.e., the oracle is a truly random function), it perfectly emulates the simulation of S.

So, we conclude that if the view of $\mathcal{A}$ in $\mathcal{H}$ is distinguishable from the view of $\mathcal{A}$ in the simulation of S, then the PPT algorithm D that, given the commitment of the PRF seed σ, can distinguish the output of PRF_σ from that of truly random function. Consider the case that D, given the commitment $c = C(\sigma)$, has oracle access to an independent $PRF_{\sigma'}$ of an independent random seed σ' or a truly random function. Due to the pseudorandomness of PRF, the output of $D(c)$ with oracle access to $PRF_{\sigma'}$ is indistinguishable from the output of $D(c)$ with oracle access to a truly random function. It implies that D, given the commitment $c = C(\sigma)$, can distinguish the output of PRF_σ and the output of $PRF_{\sigma'}$, where σ and σ' are independent random seeds. But, this violates the computational hiding property of the commitment scheme C. Specifically, given two random strings of length n, (s_0, s_1), and a commitment $c_b = C(s_b)$ for a random bit b, the algorithm D can be used to distinguish the value committed in c_b, which violates the computational hiding property of C.

Now, we consider the difference between the output of $\mathcal{H}$ and the view of $\mathcal{A}$ in real execution. Recall that, as we have shown the view of $\mathcal{A}$ in $\mathcal{H}$ is indistinguishable from that in the simulation and we have assumed Case-R2 failure does not occur in the simulation of S, Case-R2 failure can occur in $\mathcal{H}$ with at most negligible probability. Then, the difference between the output of $\mathcal{H}$ and the view of $\mathcal{A}$ in real execution lies in the witnesses used in Stage-5 of left sessions. Specifically, $\mathcal{H}$ still uses the extracted right-player secret-keys in Stage-5 of left sessions, while the honest left-player always uses its secret-key SK_L in Stage-5 of left sessions in real execution. By hybrid arguments, the difference can be reduced to violate the regular WI property of commit-then-PRZK. Note that commit-then-PRZK is itself regular WI for $\mathcal{NP}$ (actually, any commit-then-SWI is itself regular WI).

In more detail, we consider the mental experiment M_b, $b \in \{0,1\}$. On input $\{(PK_L, SK_L), (PK_R, SK_R)\}$ and public file F, and auxiliary information z to the CMIM adversary $\mathcal{A}$ [2], the mental M_b also takes as input all secret-keys corresponding to right-player public-keys in the public file F (in case the corresponding secret-keys exist). M_b runs the CMIM adversary $\mathcal{A}$ as follows:

1. M_b emulates the honest right-player of PK_R (with SK_R as the witness) in right sessions. In particular, M just sends truly random Stage-3 messages in all right sessions, and ignores knowledge-extraction of left-player secret-keys in right sessions (i.e., in case $\mathcal{A}$ successfully finishes a right session w.r.t an uncovered public-key $PK_L^{(j)}$, M_b ignores the need of secret-key extraction and just moves on);
2. For any i, j, $1 \le i \le s(n)$ and $1 \le j \le s(n)+1$, in the i-th left session w.r.t. right-player public-key $PK_R^{(j)}$, M_b emulates the honest left-player of PK_L until Stage-4 (in particular, it sets the Stage-4 message $r^{(i)}$ to be $PRF_\sigma(r_l^{(i)\prime}) \oplus \tilde{r}_r^{(i)}$), but with the following exception in Stage-5:
   - If $b = 0$, then M_b just emulates the honest left-player in Stage-5 of the left session, with SK_L as its witness.
   - If $b = 1$, M_b still emulates the simulator by using the secret-key $SK_R^{(j)}$, for which we assume it exists and M knows, as the witness in Stage-5. Specifically, it commits to $SK_R^{(j)}||0^t$ and finishes PRZK accordingly as the simulator S does.

It's easy to see that the output of M_0 is identical to the real view of $\mathcal{A}$ in real execution, and the output of M_1 is indistinguishable from the output of $\mathcal{H}$. Then, suppose the real view of $\mathcal{A}$ in real execution is distinguishable from the output of $\mathcal{H}$, by hybrid arguments we can break the regular WI of commit-then-PRZK. □

[2]Recall that, in accordance with the definition of CNMCT, z is a priori information of $\mathcal{A}$ that is independent from the public file F (in particular, PK_L and PK_R).

Now, we show that Case-R2 failure indeed occurs with negligible probability, from which the simulatability of the CNM simulation is established.

Lemma E.2 *Case-R2 failure occurs with negligible probability.*

Proof (of Lemma E.2). Suppose Case-R2 failure occurs with non-negligible probability. That is, for some polynomial $p(n)$ and infinitely many n's, with probability of $\frac{1}{p(n)}$ there exist k, i, j, $1 \leq k \leq s(n)+1$ and $1 \leq i, j \leq s(n)$, such that in the k-th simulation repetition $\mathcal{A}$ successfully finishes the i-th right session with respect to an uncovered public-key $PK_L^{(j)} \notin \mathcal{S} \cup \{PK_L\}$, furthermore, the k-th simulation repetition is the *first* one encountering Case-R2 failure and the i-th right session is the *first* successful session w.r.t. an uncovered public-key not in $\mathcal{S} \cup \{PK_L\}$ during the k-th simulation repetition, but the simulator fails in extracting the corresponding secret-key $SK_L^{(j)}$. Recall that S makes at most $s(n)+1$ simulation trials (repetitions) and each simulation trial uses fresh randomness in the proof stages; S starts knowledge-extraction whenever it encounters a successful session w.r.t. an uncovered public-key different from PK_L; Whenever Case-R2 failure occurs S aborts the whole simulation, which implies that the k-th simulation repetition is also the *last* simulation trial.

Note that, *by the AOK property of PRZK* (we can combine the k-th simulation repetition except for the Stage-5 of the i-th right session into a stand-alone knowledge prover of the PRZK), in this case the simulator still extracts some value that is determined by the statistically-binding commitment $\tilde{c}_{crs}^{(i)}$ at the start of Stage-5 of the i-th right session. According to the AOK property of PRZK, there are two possibilities for the value committed to $\tilde{c}_{crs}^{(i)}$ and extracted by S assuming Case-R2 failure.

Case-1. The value committed is the preimage of y_{1-b}. Recall that $PK_R = (y_0, y_1)$ is the simulated public-key of honest right player, with $SK_R = s_b$ for a random bit b such that $y_b = f(s_b)$.

Case-2. The value committed is the preimage of y_b.

Due to the one-wayness of the OWF f, it is easy to see that Case-1 can occur only with negligible probability. Specifically, consider the case that y_{1-b} is given to the simulator, rather than generated by the simulator itself.

Below, we show that Case-2 occurs also with negligible probability, from which Lemma E.2 is then established.

We consider the following two experiments: $E(1^n, z, PK_L, PK_R, s_b)$, where $s_b = SK_R$ and $b \in \{0,1\}$. The experiment $E(1^n, z, PK_L, PK_R, s_b)$ consists of two phases (or algorithms), denoted by E_1 and E_2: In the first phase, E_1 just runs $S(1^n, z, PK_L, PK_R, s_b)$ until S stops. Denote by $\mathcal{C}_b$ the set of extracted-keys, corresponding to public-keys in $F - \{PK_R\}$, which are extracted and used by S in its *last* simulation trial (recall that the *first* simulation repetition encountering Case-R2 failure is also the last simulation repetition). Specifically, suppose S uses $SK_R = s_b$ in the simulation and stops in the k-th simulation repetition with respect to covered-key set, denoted $\mathcal{S}_b^{(k)}$, then $\mathcal{C}_b = \mathcal{S}_b^{(k)} - \{(PK_R, SK_R)\}$. Note that $\mathcal{C}_b$ does not include (PK_R, SK_R) now. The set $\mathcal{C}_b$, the public-key file $F = F' \cup \{PK_L, PK_R\}$ and the state information τ are passed on to E_2, where (F', τ) is the output of (the key-generation stage of) the underlying CMIM adversary $\mathcal{A}(1^n, z, PK_L, PK_R)$ (run by S).

Then, in the second phase of the experiment E, a PPT algorithm $E_2(1^n, \mathcal{C}_b, F, \tau)$ runs (the proof-stage of) the CMIM adversary $\mathcal{A}(1^n, F, \tau)$ and (re)mimics the simulation of S (to be precise, S_{PROOF}) at its last simulation trial w.r.t. the set of covered-keys $\mathcal{C}_b$, but with the following exceptions (note that E_2 does not take $s_b = SK_R$ as input): (1) E_2 externally interacts with the prover of commit-then-PRZK $P(1^n, s_b)$: Whenever S needs to give a Stage-1 proof of a right session on $PK_R = (y_0, y_1)$, or needs to give a Stage-5 proof of a left session with respect to PK_R [3] on input $(PK_L, PK_R, (r_l^{(i)\prime}, \tilde{r}_r^{(i)}, r^{(i)}))$, E_2 just sets the corresponding input, i.e., PK_R or $(PK_L, PK_R, (r_l^{(i)\prime}, \tilde{r}_r^{(i)}, r_l^{(i)}))$,[4] as well as the according

[3]Note that left sessions may be with respect to the simulated public-key PK_R, i.e., the CMIM adversary may impersonate the honest right-player of PK_R in left sessions.

[4]Actually, the $\mathcal{NP}$-statements reduced from them for the $\mathcal{NP}$-Complete language for which commit-then-PRZK actually works.

left or right tag, to $P(1^n, s_b)$, and then relays messages between $P(1^n, S_b)$ and the CMIM adversary $\mathcal{A}$; (2) In case $\mathcal{A}$ successfully finishes Stage-1 of a left session with respect to an uncovered public-key not in $\mathcal{C}_b \cup \{PK_R\}$ in the run of $E_2(1^n, \mathcal{C}_b, F, \tau)$, E_2 just stops.

Now, suppose Case-2 of Case-R2 failure occurs with non-negligible probability. That is, with non-negligible probability, the simulator S aborts due to Case-R2 failure in its last simulation trial with respect to the covered public-key set $\mathcal{C}_b$, and the value committed in $\tilde{c}_{crs}^{(i)}$ (in the successful i-th right session, for some i, $1 \le i \le s(n)$, w.r.t. an uncovered public-key $PK_L^{(j)} \notin \mathcal{C}_b \cup \{PK_L, PK_R\}$ during the simulation trial w.r.t. $\mathcal{C}_b$) is the preimage of y_b. Recall that, the successful i-th right session is also the first successful session w.r.t. an uncovered public-key different from PK_L during the simulation trial w.r.t. $\mathcal{C}_b$. It is easy to see that, with also non-negligible probability, the value committed in $\tilde{c}_{crs}^{(i)}$ in the i-th right successful session (which is also the first successful session w.r.t. an uncovered public-key not in $\mathcal{C}_b \cup \{PK_L, PK_R\}$) under the run of $E_2(1^n, \mathcal{C}_b, F, \tau)$ is the preimage of y_b. We will use this fact to violate the one-left-many-right simulation/extraction of commit-then-PRZK with adaptively setting input and tag for the one left-session, where the simulator/extractor of commit-then-PRZK first commits to 0 and then runs the one-left-many-right simulator/extractor of PRZK.

Before proceeding the analysis, we first present some observations on commit-then-PRZK with restricted input selection and indistinguishable auxiliary information. Consider the following experiments: $\mathrm{EXPT}(1^n, w^b, aux^b)$, where $w^b \in \{0,1\}^n$ for $b \in \{0,1\}$. In $\mathrm{EXPT}(1^n, w^b, aux^b)$, the commit-then-PRZK for $\mathcal{NP}$ is run concurrently, and a many-left-many-right CMIM adversary $\mathcal{A}$, possessing auxiliary information aux^b and involving at most $m(n)$ left-sessions and at most $m(n)$ right-sessions simultaneously where $m(\cdot)$ is a positive polynomial, can set the inputs and tags to prover instances of left sessions with the following restriction: for any x_i, $1 \le i \le m(n)$, set by $\mathcal{A}$ for the i-th left session of commit-then-PRZK, the fixed value w^b is always a valid $\mathcal{NP}$-witness. In other words, although $\mathcal{A}$ has the power of adaptive input selection for provers, but there exists fixed witness-pair (w^0, w^1) for all inputs selected by $\mathcal{A}$. Such adversary is called *restricted* input-selecting CMIM-adversary. Denote by $trans^b$ the transcript of the experiment $\mathrm{EXPT}(1^n, w^b, aux^b)$ (i.e., the view of $\mathcal{A}$ in $\mathrm{EXPT}(1^n, w^b, aux^b)$), and by $\widehat{W}^b = \{\hat{w}_1^b, \cdots, \hat{w}_{m(n)}^b\}$ the witnesses encoded (determined) by the statistically-binding commitments (at the beginning) of successful right sessions in $trans^b$ (as in [68], in the unlikely event that a statistically-binding commitment does not uniquely determine a witness, the corresponding witness is set to be "$\perp$"); For a right session that aborts or the tag of the underlying PRZK is identical to that in one of left sessions, $\hat{w}_i^b$ is set to be a special symbol $\perp$. We want to show the following proposition:

Proposition E.1 *If the ensembles* $\{aux^0\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ *and* $\{aux^1\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ *are indistinguishable, then the ensembles* $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ *in accordance with* $\mathrm{EXPT}(1^n, w^0, aux^0)$ *and* $\{(trans^1, \widehat{W}^1)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ *in accordance with* $\mathrm{EXPT}(1^n, w^1, aux^1)$ *are also indistinguishable.*

Proof (of Proposition E.1): This is established by investigating a series of experiments.

First consider two experiments $\mathrm{EXPT}_1^n(1^n, w^b, aux^b)$, where $b \in \{0,1\}$. In $\mathrm{EXPT}_1^n(1^n, w^b, aux^b)$, a *one-left-many-right* restricted input-selecting MIM adversary $\mathcal{A}$, possessing auxiliary information aux^b, interacts with the prover instance of commit-then-PRZK in one left session and sets the input x of the left session such that $(x, w^b) \in \mathcal{R}_{\mathcal{L}}$, and concurrently interacts with many honest verifier instances on the right. From the one-many simulation/extraction SZKAOK property of PRZK (with adaptively setting input and tag for the one left session) and computational-hiding property of the underlying statistically-binding commitments, by hybrid arguments, we can conclude that if aux^0 is indistinguishable from aux^1, then $\mathcal{A}$'s views and the witnesses encoded (actually *extracted*) in the two experiments, i.e., $(trans^0, \widehat{W}^0)$ and $(trans^1, \widehat{W}^1)$), are indistinguishable.

In more details (as shown in [68]), due to the one-many simulation/extraction perfect ZKAOK property of PRZK, for any bit $b \in \{0,1\}$ $(trans^b, \widehat{W}^b)$ in $\mathrm{EXPT}_1^n(1^n, w^b, aux^b)$ is identical to $(trans^b, \widehat{W}^b)$ in a modified version of $\mathrm{EXPT}_1^n(1^n, w^b, aux^b)$, called commit($w^b$)-then-simulatedPRZK, in which the one left-session and many right-sessions are emulated by a PPT algorithm (i.e., the one-left-many-right simulator/extractor guaranteed by PRZK) with witness extraction for successful right-sessions of different tags (but the witness w^b is still committed to the statistically-binding commitment of the left session). Then, for this experiment, due to the computational hiding property of the statistically-binding commitment scheme used in commit-then-PRZK, $(trans^b, \widehat{W}^b)$ of the commit(w^b)-then-simulatedPRZK experiment is computationally indistinguishable from that of the commit(0)-then-simulatedPRZK experiment in which "0" (rather than w^b) is committed to the statistically-binding commitment of the one left session. Note that the commit(0)-then-simulatedPRZK experiment can be performed by a merely PPT algorithm. The reader is referred to [68] for more details of the hybrid arguments of this step. Here, we point out that the hybrid arguments of [68] is actually w.r.t. a *strengthened* version of commit-then-PRZK, which is referred as *signed* commit-then-PRZK here. Roughly speaking, the tag of the underlying PRZK is set to be the public-key of a signature scheme, and the protocol transcript of commit-then-PRZK is in turn signed by the prover. Some advantages of signed commit-then-PRZK are: it can work for tags of length $poly(n)$ (rather than $O(n)$ as required by the underlying PRZK), and it can satisfy some stronger non-malleability requirements w.r.t. session transcripts (rather than only session tags or inputs). We note this signature-based trick is unnecessary for our purpose in this work. In particular, the tags of the underlying PRZK in our CNMCT constructions are indeed of length $O(n)$, and our CNM definitions (for ZK and CT) are based on the normal formulation approach that is not w.r.t. session transcripts.[5]

Now we consider the following two experiments: $\text{EXPT}(1^n, w, aux^b)$, where $b \in \{0,1\}$ and $w \in \{w^0, w^1\}$. In $\text{EXPT}(1^n, w, aux^b)$, a many-left-many-right restricted input-selecting MIM adversary $\mathcal{A}$, possessing auxiliary information aux^b, interacts concurrently with many prover instances on the left (such that w is always a witness for inputs selected adaptively by $\mathcal{A}$ for left sessions), and interacts with many honest verifier instances on the right. Then, the indistinguishability between the ensembles $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ and $\{(trans^1, \widehat{W}^1)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ is direct from the indistinguishability between $\{aux^0\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ and $\{aux^1\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ and the adaptive one-left-many-right simulation-extractability of PRZK. Specifically, this is derived by a simple reduction to the above one-left-many-right case. Note that according to the definition of indistinguishability between *ensembles*, (w, aux^0) and (w, aux^1) are indistinguishable. *Actually,* (w^0, w^1, aux^0) *and* (w^0, w^1, aux^1) *are indistinguishable.* Also, note that all sessions in $\text{EXPT}(1^n, w, aux^b)$ can be emulated internally by a PPT algorithm given (w, aux^b).

In more details, we consider the experiments $\text{EXPT}_1^n(1^n, w, aux^b)$, where $b \in \{0,1\}$ and $w \in \{w^0, w^1\}$. In the experiment $\text{EXPT}_1^n(1^n, w, aux^b)$, a one-left-many-right MIM adversary $\mathcal{A}'$ that on auxiliary input (w, aux^b) mimics the CMIM adversary $\mathcal{A}$ (of the auxiliary input aux^b) in the above experiment $\text{EXPT}(1^n, w, aux^b)$, with the following modifications: all left-sessions except for the first left-session are perfectly emulated by $\mathcal{A}'$ by using w as the witness, and $\mathcal{A}'$ externally interacts with the commit-then-PRZK prover in the first left-session; $\mathcal{A}'$ outputs a simulated view of $\mathcal{A}$ that is identical to the view of $\mathcal{A}$ in the experiment $\text{EXPT}(1^n, w, aux^b)$. By the adaptive one-left-many-right simulation-extractability of PRZK, the view of $\mathcal{A}$ and the corresponding witnesses encoded by the statistically-binding commitments under the run of $\mathcal{A}'(w, aux^b)$ are indistinguishable from the outputs of the PPT simulator/extractor guaranteed by PRZK, with auxiliary input (w, aux^b), in the commit(0)-then-simulatedPRZK experiment. As (w, aux^0) and (w, aux^1) are indistinguishable, we conclude that $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ in accordance with the experiment $\text{EXPT}(1^n, w, aux^0)$ and $\{(trans^1, \widehat{W}^1)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ in accordance with $\text{EXPT}(1^n, w, aux^1)$ are indistinguishable.

We return back to investigate the experiments: $\text{EXPT}(1^n, w^b, aux^b)$ with respect to many-left-many-right restricted input-selecting MIM adversary $\mathcal{A}$. Firstly, the distribution ensemble of $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ in accordance with $\text{EXPT}(1^n, w^0, aux^0)$ and the distribution ensemble of $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ *in accordance with* $\text{EXPT}(1^n, w^0, aux^1)$ are indistinguishable, if $\{aux_0\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ and $\{aux_1\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ are indistinguishable, where $\text{EXPT}(1^n, w^0, aux^1)$ denotes a hybrid experiment in which the CMIM adversary possesses auxiliary in-

[5]The extension to session-transcript based formulations and protocol implementations of CNMCT and CNMZK are left for future explorations.

formation aux^1 while concurrently interacting on the left with many prover instances of the fixed witness w^0. Then, by a simple hybrid argument to the one-left-many-right case, we get that the distribution ensemble $\{(trans^0, \widehat{W}^0)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ in accordance with EXPT$(1^n, w^0, aux^1)$ is indistinguishable from the distribution ensemble of $\{(trans^1, \widehat{W}^1)\}_{n\in N, w^0\in\{0,1\}^n, w^1\in\{0,1\}^n}$ in accordance with EXPT$(1^n, w^1, aux^1)$. In more detail, if the above ensembles are distinguishable, then the difference can be reduced, by hybrid arguments, to the difference of witnesses used in only one left session. Note that, all sessions other than the one left session can be emulated internally by a PPT algorithm given (w^0, w^1, aux^1). We remark that, here, posing the *restricted* input selection requirement in EXPT$(1^n, w^b, aux^b)$, i.e., the fixed w_0 and w_1 are always the valid witnesses to all statements set by the CMIM adversary for left-sessions, is critical for the above hybrid arguments to go through.

Proposition E.1 follows. □

Now, we return back to the experiment $E(1^n, z, PK_L, PK_R, s_b)$ for finishing the proof of Lemma E.2. We first prove that $\{\mathcal{C}_0\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ is indistinguishable from $\{\mathcal{C}_1\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ according to the analysis of Proposition E.1, where $\mathcal{C}_b$, $b \in \{0,1\}$, is the set of extracted-keys (corresponding to public-keys in $F - \{PK_R\}$) that is used by the simulator S in its last simulation repetition. Equivalently, $\mathcal{C}_b$ is generated by E_1 and is passed on to E_2. Note that $s_b = SK_R$ is the simulated secret-key used by S (equivalently, E_1). For presentation simplicity, in the following description we simply refer to S, E_1 and E_2 as $S(1^n, s_b)$, $E_1(1^n, s_b)$ and $E_2(1^n, \mathcal{C}_b)$. Actually, we can show that for any k, $1 \le k \le s(n)+1$, if the distribution ensemble of the set of extracted-keys used in the $(k-1)$-th simulation repetition of $S(1^n, s_0)$ using $SK_R = s_0$, denoted $\{\mathcal{C}_0^{k-1}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$, is indistinguishable from that of $\{\mathcal{C}_1^{k-1}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ (the set of extracted-keys used in the $(k-1)$-th simulation repetition of $S(1^n, s_1)$), then the distribution ensembles of $\{\mathcal{C}_0^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ and $\{\mathcal{C}_1^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ are also indistinguishable.

We consider another PPT algorithm $\hat{S}(1^n, \mathcal{C}_b^{k-1})$ that mimics $E_2(1^n, \mathcal{C}_b^{k-1})$ (with externally interacting with the commit-then-PRZK prover $P(1^n, s_b)$) but with the following modifications: the interactions of Stage-1 of left-sessions and Stage-5 of right-sessions, in which the underlying CMIM adversary $\mathcal{A}$ serves as the prover of commit-then-PRZK, are relayed by $\hat{S}$ between the underlying CMIM adversary $\mathcal{A}$ and external commit-then-PRZK verifiers (who actually just send random coins, as PRZK is actually *public-coin*). We remark that the run of $\hat{S}(1^n, \mathcal{C}_b^{k-1})$ actually amounts to the experiment EXPT$(1^n, w^b, aux^b)$ defined in Proposition E.1, where $2s(n)$ amounts to $m(n)$ as $\hat{S}$ can involve at most $2s(n)$ sessions in each (left or right) CMIM interaction part, s_b amounts to w^b, $\mathcal{C}_b^{k-1}$ amounts to aux^b and the interactions with the at most $2s(n)$ instances of the commit-then-PRZK prover $P(1^n, s_b)$ amount to the left-sessions and the interactions between $\hat{S}$ (actually $\mathcal{A}$) and the at most $2s(n)$ instances of the commit-then-PRZK verifier amount to right-sessions. Here, a point of worthy noting is: though commit-then-PRZK is composed with other interactions (say, the interactions at Stage-2, Stage-3 and Stage-4), all interactions other than the interactions with the prover $P(s_b)$ of commit-then-PRZK (i.e., the left-sessions of $\hat{S}$) can be internally emulated by $\hat{S}$, though Stage-1 interactions of left-sessions and Stage-5 interactions of right-sessions (which correspond to the right-sessions of $\hat{S}$ and are just public coins) are not internally emulated by $\hat{S}$. By applying Proposition E.1, we get that if the ensembles $\{\mathcal{C}_0^{k-1}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ and $\{\mathcal{C}_1^{k-1}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ are distinguishable, $\{\mathcal{C}_0^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ and $\{\mathcal{C}_1^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ are also distinguishable. Finally, note that $\mathcal{C}_0^0$ and $\mathcal{C}_1^0$ (the set of extracted-keys corresponding to $F - \{PK_R\}$ at the beginning of the simulation) are identical, i.e., both of them are the empty set. By inductive steps, we get that the distribution ensembles of $\{\mathcal{C}_0^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ and $\{\mathcal{C}_1^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ are indistinguishable for any k, $1 \le k \le s(n)+1$. *Here, we note that the above analysis (for showing the indistinguishability between* $\{\mathcal{C}_0^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$ *and* $\{\mathcal{C}_1^{k}\}_{n\in N, s_0\in\{0,1\}^n, s_1\in\{0,1\}^n}$*, for any* k*,* $1 \le k \le s(n)+1$*) works also for the case that* $\mathcal{C}_0^0 = \mathcal{C}_1^0 = \{s_\alpha\}$*, where* $\alpha \in \{0,1\}$ *and* $s_\alpha \in \{s_0, s_1\}$ *is a fixed value and is independent of* s_b*. This property will be used in the subsequent analysis of secret-key independence.*

But, suppose Case-2 of Case-R2 failure occurs with non-negligible probability. Then, with also non-negligible probability, the value committed to the statistically-binding commitment (at the beginning) of a (actually the first) successful right-session of commit-then-PRZK w.r.t. an uncovered public-key not in $\mathcal{C}_b \cup \{PK_L, PK_R\}$ under the run of the CMIM algorithm $\hat{S}$ (equivalently, $E_2(1^n, \mathcal{C}_b)$), with auxiliary input $\mathcal{C}_b$ where $b \in \{0,1\}$ and $\mathcal{C}_0$ and $\mathcal{C}_1$ are indistinguishable, is the preimage of y_b. Suppose this right-session is the i-th right-session run by $\hat{S}(1^n, s_b)$, $1 \le i \le 2s(n)$, it can be directly checked that, with overwhelming probability, the tag used by Stage-5 of this i-th right session, denoted $(PK_L^{(j)}, r_r^{(i)}, \tilde{r}^{(i)})$ where $PK_L^{(j)} \notin \mathcal{C}_b \cup \{PK_L, PK_R\}$ and $r_r^{(i)}$ is a random n-bit string, must be different from the tags used by the prover $P(1^n, s_b)$ of commit-then-PRZK. Recall that the tags of Stage-1 of right sessions (*run by* $P(s_b)$) is of the form $(\cdot, y_0, y_1)$ and the tags of Stage-5 of left sessions (*run by* $P(s_b)$) is of the form $(PK_L, \cdot, \cdot)$. This means that, by concurrently interacting with the prover $P(s_b)$ of commit-then-PRZK in left-sessions and with the commit-then-PRZK verifier instances in the right-sessions, $\hat{S}(1^n, \mathcal{C}_b)$ can successfully commit the preimage of y_b in a successful right session that is of a tag different from all the tags of the left-session interactions with $P(1^n, s_b)$ and is actually the first right-session w.r.t an uncovered public-key not in $\mathcal{C}_b \cup \{PK_R, PK_L\}$, which violates Proposition E.1. This shows that Case-2 of Case-R2 failure can occur also with negligible probability. Thus, Case-R2 failure can occur with at most negligible probability. This finishes the proof of Lemma E.2, from which the simulatability of the CNM simulation depicted in Figure 3 is then established. □

Next, before proceeding the analysis of the property of strategy-restricted and pre-definable randomness, we first investigate the property of secret-key independence which is essentially implied by the above analysis of Lemma E.2 and Proposition E.1.

- Secret-key independence

Specifically, we need to show that $\Pr[\mathcal{R}(SK_R, str, sta) = 1]$ is negligibly close to $\Pr[\mathcal{R}(SK'_R, str, sta) = 1]$ for any polynomial-time computable relation $\mathcal{R}$. In more details, for any pair (s_0, s_1) in the (simulated right-player) key-generation stage, denote by (str^b, sta^b) the output of $S(1^n, s_b)$ when it is using $SK_R = s_b$. Then, $\Pr[\mathcal{R}(SK, str, sta) = 1] = \frac{1}{2}\Pr[\mathcal{R}(s_0, str^0, sta^0) = 1 | S$ uses $SK_R = s_0$ in generating $(str^0, sta^0)] + \frac{1}{2}\Pr[\mathcal{R}(s_1, str^1, sta^1) = 1 | S$ uses $SK_R = s_1$ in generating $(str^1, sta^1)]$, and $\Pr[\mathcal{R}(SK'_R, str, sta) = 1] = \frac{1}{2}\Pr[\mathcal{R}(s_0, str^1, sta^1) = 1 | S$ uses $SK_R = s_1$in generating $(str^1, sta^1)] + \frac{1}{2}\Pr[\mathcal{R}(s_1, str^0, sta^0) = 1 | S$ uses $SK_R = s_0$ in generating $(str^0, sta^0)]$. Suppose the secret-key independence property does not hold, it implies that there exists a bit $\alpha \in \{0,1\}$ such that the difference between $\Pr[\mathcal{R}(s_\alpha, str^0, sta^0) = 1 | S$ uses s_0 in generating $(str^0, sta^0)]$ and $\Pr[\mathcal{R}(s_\alpha, str^1, sta^1) = 1 | S$ uses s_1 in generating $(str^1, sta^1)]$ is non-negligible. It implies that (s_α, str^0, sta^0) and (s_α, str^1, sta^1) are distinguishable. But, note that the analysis of Lemma E.2 and Proposition E.1 has already established that the distribution ensembles of $\{S(1^n, s_0) = (str^0, sta^0)\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ and $\{S(1^n, s_1) = (str^1, sta^1)\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ are indistinguishable. Specifically, the distribution ensembles of the sets of extracted-keys corresponding to the public-keys in $F - \{PK_R\}$, $\{\mathcal{C}_0\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ and $\{\mathcal{C}_1\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ used by $S(1^n, s_b)$ for $b \in \{0,1\}$ in the last simulation repetition, are indistinguishable, and then the indistinguishability between the ensembles $\{(str^0, sta^0)\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ and $\{(str^1, sta^1)\}_{n \in N, s_0 \in \{0,1\}^n, s_1 \in \{0,1\}^n}$ are from Proposition E.1.

- Strategy-restricted and predefinable randomness

Now, we proceed to show the strategy-restricted and predefinable randomness property of the CNM simulator S depicted in Figure 3. Denote by $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots R_L^{(s(n))}\}$ the coin-tossing outputs of the $s(n)$ left sessions in str (i.e., the first output of S), and by $sta_L = \{sta_L^{(1)}, sta_L^{(2)}, \cdots, sta_L^{(s(n))}\}$ the state information corresponding to R_L included in sta (i.e., the second output of S). Similarly, denote by $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots R_R^{(s(n))}\}$ the coin-tossing outputs of the $s(n)$ right sessions in str, and by $sta_L = \{sta_R^{(1)}, sta_R^{(2)}, \cdots, sta_R^{(s(n))}\}$ the state information for R_R. We want to show that, with overwhelming probability, both the distribution of (R_L, sta_L) and that of (R_R, sta_R) are identical to that of $\mathcal{M}_{CRS}^{s(n)}(1^n)$. Recall that, $(\{r_1, r_2, \cdot, r_{s(n)}\}, \{\tau_{r_1}, \tau_{r_2}, \cdots, \tau_{r_{s(n)}}\}) \longleftarrow \mathcal{M}_{CRS}^{s(n)}(1^n)$ denotes the output of the experiment of running $\mathcal{M}_{CRS}(1^n)$ *independently* $s(n)$ times.

Note that, according to the CNM simulation described in Figure 3, for any i, $1 \le i \le s(n)$, the output of the i-th left session, i.e., $R_L^{(i)}$, in the simulation is always $S_L^{(i)}$ and $sta_L^{(i)}$ is always $\tau_L^{(i)}$, where

$(S_L^{(i)}, \tau_L^{(i)})$ is the output of an independent run of $\mathcal{M}_{CRS}(1^n)$. It is directly followed that the distribution of (R_L, sta_L) is identical to that of $\mathcal{M}_{CRS}^{s(n)}(1^n)$.

The complicated point here is to show that, with overwhelming probability, the distribution of (R_R, sta_R) is also identical to that of $\mathcal{M}_{CRS}^{s(n)}(1^n)$. According to the CNM simulation depicted in Figure 3, if we can prove that, with overwhelming probability, for any i, $1 \leq i \leq s(n)$, the coin-tossing output of the successful i-th right session $R_R^{(i)}$ is either $S_R^{(i)}$ or $R_L^{(k)} = S_L^{(k)}$ for some k, $1 \leq k \leq s(n)$; furthermore, any left-session output $S_L^{(k)}$ can be the coin-tossing output for at most *one* successful right session (which implies the coin-tossing outputs of successful right sessions are independent), then the distribution of (R_R, sta_R) is also identical to that of $\mathcal{M}_{CRS}^{s(n)}(1^n)$. In the following description, for presentation simplicity, we sometimes omit some unlikely events occurring with negligible probability.

For any i, $1 \leq i \leq s(n)$, we consider the successful i-th right session with respect to a public-key $PK_L^{(j)}$. As we have shown that Case-R2 failure occurs with negligible probability, we get $PK_L^{(j)} \in \mathcal{C}_b \cup \{PK_R, PK_L\}$, where $\mathcal{C}_b$ is the set of extracted-keys (corresponding to public-keys in $F - \{PK_R\}$) used by $S(s_b)$ in its last simulation repetition.

We first observe that, if $PK_L^{(j)} = PK_L$ then with overwhelming probability the tag of Stage-5 of the successful i-th right session must be identical to that of Stage-5 of a left session simulated by the simulator S. Recall that all the Stage-5 tags of right sessions are different strings, as they contain random Stage-3 strings sent by the simulator. This means that Stage-5 tags of right sessions are also different from Stage-1 tags of right sessions simulated by S (note that all Stage-1 tags of right sessions consist of the fixed PR_R). Now, suppose the Stage-5 tag of the successful i-th right session is also different from the Stage-5 tags of all left sessions simulated by S, then it implies that the tag used by the CMIM adversary for Stage-5 of the i-th right session is different from all tags used by the simulator (particularly, the prover $P(s_b)$ of commit-then-PRZK run by $E_2(1^n, \mathcal{C}_b)$ or $\hat{S}(1^n, \mathcal{C}_b)$ in the analysis of Lemma E.2).

By the AOK property of PRZK, it implies that the value committed to $\tilde{c}_{crs}^{(i)}$ (sent by $\mathcal{A}$ in Stage-5 of the i-th right session) can be extracted. We consider the possibilities of the value committed to $\tilde{c}_{crs}^{(i)}$:

- By the one-wayness of y_{1-b} the value committed cannot be the preimage of y_{1-b};
- According to the analysis of Lemma E.2, the value also cannot be the preimage of y_b.

Thus, the value committed (that can be extracted) will be the secret-key of PK_L, which however violates the one-wayness of PK_L as the simulator never knows and uses the secret-key of PK_L in its simulation. Thus, we conclude that, if a successful right session is w.r.t. PK_L, the tag used by $\mathcal{A}$ for commit-then-PRZK of Stage-5 must be identical to that of one left-session simulated by S. As the Stage-5 tag consists of the coin-tossing output, i.e., the Stage-4 message, this means that the coin-tossing output of the i-th right session must be $R_L^{(k)} = S_L^{(k)}$ for some k, $1 \leq k \leq s(n)$.

Now, we consider the case $PK_L^{(j)} \neq PK_L$ but $PK_L^{(j)} \in \mathcal{C}_b \cup \{PK_R\}$. In this case, S has already learnt the corresponding secret-key $SK_L^{(j)}$. Now, suppose the coin-tossing output of the successful i-th right session is neither $S_R^{(i)}$ nor $R_L^{(k)} = S_L^{(k)}$ for all k, $1 \leq k \leq s(n)$. This implies that the Stage-5 tag used by $\mathcal{A}$ in the successful i-th right session is different from Stage-5 tags of all left sessions [6] as well as the Stage-1 tags of all right sessions simulated by S. Again, by the AOK property of PRZK, we consider the value committed to $\tilde{c}_{crs}^{(i)}$: According to the simulation of S, it always sets Stage-3 message $r_r^{(i)}$ of right session to be $PRF_{SK_L^{(j)}}(\tilde{r}_l^{(i)\prime}) \oplus S_R^{(i)}$, where $\tilde{r}_l^{(i)\prime}$ is the Stage-2 message of the i-th right session sent by the CMIM adversary $\mathcal{A}$. Suppose the coin-tossing output of the successful i-th right session is not $S_R^{(i)}$, then (by the AOK of PRZK) the value committed to $\tilde{c}_{crs}^{(i)}$ cannot be $SK_L^{(j)}$, as otherwise (with overwhelming probability) the $\mathcal{NP}$-statement to be proved by PRZK in Stage-5 of the i-th right session is false. This means that the value committed to $\tilde{c}_{crs}^{(i)}$ will be the preimage of either y_{1-b} or y_b.

[6] Note that all Stage-5 tags of left sessions are of the form $(PK_L, \cdot, \cdot)$, and the Stage-5 tag of the successful i-th right session is of the form $(PK_L^{(j)}, \cdot, \cdot)$ for $PK_L^{(j)} \neq PK_L$.

But, each case reaches the contradiction: committing to the preimage of y_{1-b} is impossible due to the one-wayness of y_{1-b}; committing to the preimage of y_b violates the one-left-many-right non-malleability of PRZK as demonstrated in the analysis of Lemma E.2. So, we conclude that, with overwhelming probability, for any successful right session the coin-tossing output is either the independent value $S_R^{(i)}$ or $S_L^{(k)}$ for some k, $1 \leq k \leq s(n)$ (i.e., the coin-tossing output of one left session).

To finally establish the property of strategy-restricted and predefinable randomness, we need to further show, for any $S_L^{(k)}$ it can occur as Stage-4 message (i.e., the coin-tossing output) for *at most one successful right session*. Suppose there are i_0, i_1, $1 \leq i_0 \neq i_1 \leq s(n)$, such that both of the i_0-th right session and the i_1-th right session are successful with the same Stage-4 message $S_L^{(k)}$. Recall that the Stage-5 tag of each of the two right sessions includes the same $S_L^{(k)}$ as well as a random Stage-3 message sent by the simulator; Also note that the $S_L^{(k)}$ can appear as a part of Stage-5 tag, as well as coin-tossing output, for at most one left session, as all coin-tossing outputs (i.e., Stage-4 messages) of left sessions are independent random strings output by $\mathcal{M}_{CRS}$. This implies that, with overwhelming probability, there must exist a bit b such that the Stage-5 tag of the i_b-th right session is different from all Stage-5 tags of left sessions (run by the simulator) and Stage-1 tags of right sessions (run by the simulator). According to above clarifications and analysis, with overwhelming probability, the (left-player) public-key $PK_L^{(j)}$ used by $\mathcal{A}$ in the i_b-th successful right session is *covered* and is *not* PK_L (as any right-session w.r.t. PK_L is of tag identical to that of one left-session), and the value committed in $\tilde{c}_{crs}^{(i_b)}$ is neither the secret-key of the *covered* public-key $PK_L^{(j)}$ (as, otherwise, the $\mathcal{NP}$-statement successfully proved by PRZK in the Stage-5 of the i_b-th right-session is actually false) nor the preimage of y_{1-b} (due to the one-wayness of f); Also, the value committed cannot be the preimage of y_b in accordance with the analysis of Lemma E.2. Contradiction is reached in either case.

The proof of Theorem 5.1 is finished. □