Concurrent Zero Knowledge without Complexity Assumptions

Daniele Micciancio[*] Shien Jin Ong[†] Amit Sahai[‡] Salil Vadhan[§]

August 23, 2005

Abstract

We provide *unconditional* constructions of *concurrent* statistical zero-knowledge proofs for a variety of non-trivial problems (not known to have probabilistic polynomial-time algorithms). The problems include Graph Isomorphism, Graph Nonisomorphism, Quadratic Residuosity, Quadratic Nonresiduosity, a restricted version of Statistical Difference, and approximate versions of the (**coNP** forms of the) Shortest Vector Problem and Closest Vector Problem in lattices.

For some of the problems, such as Graph Isomorphism and Quadratic Residuosity, the proof systems have provers that can be implemented in polynomial time (given an **NP** witness) and have $\tilde{O}(\log n)$ rounds, which is known to be essentially optimal for black-box simulation.

To our best of knowledge, these are the first constructions of concurrent zero-knowledge protocols in the asynchronous model (without timing assumptions) that do not require complexity assumptions (such as the existence of one-way functions).

Keywords: concurrent zero-knowledge proofs, problem-dependent commitments, quadratic residuosity, graph isomorphism, statistical difference.

[*]Computer Science and Engineering Department, University of California, San Diego, 9500 Gilman Drive, Mail Code 0404, La Jolla, CA 92093-0404. E-mail: daniele@cs.ucsd.edu. Supported by NSF grant 0313241 and an Alfred P. Sloan Research Fellowship.

[†]Division of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA 02138. E-mail: shienjin@eecs.harvard.edu. Supported by ONR grant N00014-04-1-0478.

[‡]Computer Science Department, University of California, Los Angeles, 3731E Boelter Hall, Los Angeles, CA 90095. E-mail: sahai@cs.ucla.edu. Part of this research was done while author was at Princeton University. Supported by NSF ITR and Cybertrust programs, an equipment grant from Intel, and an Alfred P. Sloan Foundation Fellowship.

[§]Division of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA 02138. E-mail: salil@eecs.harvard.edu. URL: `http://eecs.harvard.edu/~salil/`. Supported by NSF grants CNS-0430336 and CCR-0205423.

1 Introduction

In the two decades since their introduction [GMR89], zero-knowledge proofs have taken on a central role in the study of cryptographic protocols, both as a basic building block for more complex protocols and as a testbed for understanding important new issues such as composability (*e.g.,* [GK96]) and concurrency (*e.g.,* [DNS98]). The "classic" constructions of zero-knowledge proofs came primarily in two flavors. First, there were direct constructions of zero-knowledge proofs for specific problems, such as Quadratic Residuosity [GMR89] and Graph Isomorphism [GMW91]. Second, there were general constructions of zero-knowledge proofs for entire classes of problems, such as all of **NP** [GMW91].[1] Both types of results have played an important role in the development of the field.

The general results of the second type show the wide applicability of zero knowledge, and are often crucial in establishing general feasibility results for other cryptographic problems, such as secure multiparty computation [Yao86, GMW91] and CCA-secure public-key encryption [NY90, DDN01, Sah99]. However, they typically are too inefficient to be used in practice. The specific results of the first type are often much more efficient, and are therefore used in (or inspire) the construction of other efficient cryptographic protocols, *e.g.,* identification schemes [FFS88] and again CCA-secure public-key encryption [CS98, ES02, CS04]. Moreover, the specific constructions typically do not require any unproven complexity assumptions (such as the existence of one-way functions), and yield a higher security guarantee (such as *statistical* zero-knowledge proofs).[2] The fact that the proof systems are unconditional is also of conceptual interest, because they illustrate the nontriviality of the notion of zero knowledge even to those who are unfamiliar with (or who do not believe in the existence of) one-way functions.[3]

Concurrent zero knowledge. In recent years, a substantial effort has been devoted to understanding the security of cryptographic protocols when many executions are occurring concurrently (with adversarial scheduling). As usual, zero-knowledge proofs led the way in this effort, with early investigations of concurrency for relaxations of zero knowledge dating back to Feige's thesis [Fei90], and the recent interest being sparked by the work of Dwork, Naor, and Sahai [DNS98], which first defined the notion of concurrent zero knowledge. Research on concurrent zero knowledge has been very fruitful, with a sequence of works leading to essentially tight upper and lower bounds on round complexity for black-box simulation [RK99, KPR98, KP01, Ros00, CKPR03, PRS02], and partly motivating the first non-black-simulation zero-knowledge proof [Bar01]. However, these works are primarily of the *second* type mentioned above. That is, they are general feasibility results, giving protocols for all of **NP**. As a result, these protocols are fairly inefficient (in terms of computation and communication), rely on unproven complexity assumptions, and only yield computational zero knowledge (or, alternatively, computational soundness).

There have been a couple of works attempting to overcome these deficiencies. Di Crescenzo [DiC00] gave unconditional constructions of concurrent zero-knowledge proofs in various timing models.

[1] See the textbook [Gol01] and survey [Gol02] by Oded Goldreich for a thorough introduction to zero-knowledge proofs.

[2] Of course, this partition into two types of zero-knowledge protocols is not a precise one. For example, there are some efficient zero-knowledge proofs for specific problems that use complexity assumptions (*e.g.,* [GMR98] and there are some general results that are unconditional (*e.g.,* [Oka00, GSV98, Vad04]).

[3] It should be noted that the results of [Ost91, OW93] show that the existence of a zero-knowledge proof for a problem outside **BPP** implies some weak form of one-way function. Still, appreciating something like the perfect zero-knowledge proof system for Graph Isomorphism [GMW91] only requires believing that there is no *worst-case* polynomial-time algorithm for Graph Isomorphism, as opposed to appreciating notions of average-case complexity as needed for standard one-way functions.

That is, his protocols assume that the honest parties have some synchronization and may employ delays in the protocol, and thus do not work in the standard, asynchronous model (and indeed he states such a strengthening as an open problem). Micciancio and Petrank [MP03] gave an efficient (in terms of computation and communication) transformation from honest-verifier zero-knowledge proofs to concurrent zero-knowledge proofs. However, their transformation relies on the Decisional Diffie–Hellman assumption, and yields only computational zero knowledge.

Our Results. We give the first unconditional constructions of concurrent zero-knowledge proofs in the standard, asynchronous model. Our proof systems are statistical zero knowledge and statistically sound (i.e. they are interactive proofs, not arguments [BCC88]). Specifically, our constructions fall into two categories:

1. Efficient proof systems for certain problems in **NP**, including Quadratic Residuosity, Graph Isomorphism and a restricted form of quadratic nonresiduosity for Blum integers, which we call Blum Quadratic Nonresiduosity. These proof systems all have prover strategies that can be implemented in polynomial time given an **NP** witness and have $\widetilde{O}(\log n)$ rounds, which is essentially optimal for black-box simulation [CKPR03].
2. Inefficient proof systems for other problems, most of which are not known to be in **NP**. These include Quadratic Nonresiduosity, Graph Nonisomorphism, the approximate versions of the complements of the Closest Vector Problem and Shortest Vector Problem in lattices, and a restricted version of Statistical Difference (the unrestricted version is complete for statistical zero knowledge [SV03]). These proof systems, with the exception for Quadratic Nonresiduosity, have a polynomial number of rounds, and do not have polynomial-time prover strategies. These deficiencies arise from the fact that our construction begins with a public-coin, honest-verifier zero-knowledge proof for the problem at hand, and the only such proofs known for the problems listed here have a polynomial number of rounds and an inefficient prover strategy.

Techniques. One of the main tools for constructing zero-knowledge proofs are commitment schemes, and indeed the only use of complexity assumptions in the construction of zero-knowledge proofs for all of **NP** [GMW91] is to obtain a commitment scheme (used by the prover to commit to the **NP** witness, encoded as, *e.g.,* a 3-coloring of a graph). Our results rely on a relaxed notion of commitment, called a *problem-dependent commitment scheme*, which is implicit in [BMO90] and formally defined in [IOS97, MV03, Vad04]. Roughly speaking, for a language L (or, more generally, a promise problem), a problem-dependent commitment scheme for L is a commitment protocol where the sender and receiver algorithms also depend on the instance x. The security requirements of the protocol are relaxed so that the hiding property is only required when $x \in L$, and the binding property is only required when $x \notin L$ (or vice-versa).

As observed in [IOS97], many natural problems, such as Graph Isomorphism and Quadratic Residuosity, have simple, unconditional problem-dependent commitment schemes. This is useful because in many constructions of zero-knowledge proofs (such as that of [GMW91]), the hiding property of the commitment scheme is only used to establish the zero-knowledge property and the binding property of the commitment scheme is only used to establish soundness. Since, by definition, the zero-knowledge property is only required when the input x is in the language, and the soundness condition is only required when x is not in the language, it suffices to use a problem-dependent commitment scheme. Specifically, if a language $L \in \mathbf{NP}$ (or even $L \in \mathbf{IP}$) has a problem-dependent commitment scheme, then L has a zero-knowledge proof [IOS97] (see also [MV03, Vad04]).

Existing constructions of *concurrent* zero-knowledge proofs [KPR98, CKPR03, PRS02] also rely on commitment schemes (and this is the only complexity assumption used). Thus it is natural to try to use problem-dependent commitments to construct them. However, these protocols use commitments not only from the prover to the verifier, but also from the verifier to the prover. Naturally, for the latter type of commitments, the roles of the hiding and binding property are reversed from the above — the hiding property is used to prove soundness and the binding property is used to prove (concurrent) zero knowledge. Thus, it seems that we need not only a problem-dependent commitment as above, but also one where the security properties are reversed (i.e. binding when $x \in L$, and hiding when $x \notin L$).

Our first observation is that actually we only need to implement the commitment schemes from the verifier to the prover. This is because the concurrent zero-knowledge proof system of Prabhakaran, Rosen and Sahai [PRS02] is constructed by a general compiler that converts *any* public-coin zero-knowledge proof into a concurrent zero-knowledge proof, and this compiler only uses commitments from the verifier to the prover. (Intuitively, the verifier commits to its messages in an initial "preamble" stage, which is designed so as to allow concurrent simulation.) Since all the problems we study are unconditionally known to have public-coin zero-knowledge proofs, we only need to implement the compiler. So we are left with the task finding problem-dependent commitments that are binding when $x \in L$ and hiding when $x \notin L$. Thus, for the rest of the paper, we use this as our definition of problem-dependent commitment.

This idea works directly for some problems, such as Graph Nonisomorphism and Quadratic Nonresiduosity. For these problems, we have problem-dependent commitments with the desired security properties, and thus we can directly use these commitments in the compiler of [PRS02]. Unfortunately, for the complement problems, such as Graph Isomorphism and Quadratic Residuosity, we only know of problem-dependent commitments that are hiding when $x \in L$, and binding when $x \notin L$.

Thus, for some of our results, we utilize a more sophisticated variant of problem-dependent commitments, due to Bellare, Micali, and Ostrovsky [BMO90]. Specifically, they construct something like a problem-dependent commitment scheme for the Graph Isomorphism problem, but both the hiding and binding properties are non-standard. For example, the binding property is as follows: they show that if $x \in L$ and the sender can open a commitment in two different ways, then it is possible for the sender to extract an **NP** witness for $x \in L$. Thus we call these *witness-binding commitments*. Intuitively, when we use such commitments, we prove concurrent zero knowledge by the following case analysis: either the verifier is bound to its commitments, in which case we can simulate our proof system as in [PRS02], *or* the simulator can extract a witness, in which case it can be simulated by running the honest prover strategy. In reality, however, the analysis does not break into such a simple case analysis, because the verifier may break the commitment scheme in the middle of the protocol. Thus we require that, in such a case, an already-begun simulation can be "continued" once we are given an **NP** witness. Fortunately, the classic (stand-alone) proof systems for Graph Isomorphism and Quadratic Residuosity turn out to have the needed "witness-completable simulation" property.

An additional contribution of our paper is to provide abstractions and generalizations of all of the above tools that allow them to be combined in a modular way, and may facilitate their use in other settings. First, we show how the "preamble" of the Prabhakaran–Rosen–Sahai concurrent zero-knowledge proof [PRS02] can be viewed as a way to transform any commitment scheme into one that is "concurrently extractable," in the sense that we are able to simulate the concurrent execution of many sessions between an adversarial sender and the honest receiver in a way that allows us to extract the commitments of the sender in every session. This may be useful in constructing other concurrently secure protocols (not just proof systems). Second, we provide general definitions of witness-binding commitment schemes as well as witness-completable zero-knowledge proofs as possessed by GRAPH ISOMORPHISM and QUADRATIC RESIDUOSITY and as discussed above.

Perspective. The recent works of Micciancio and Vadhan [MV03] and Vadhan [Vad04] hypothesized that every problem that has a statistical (resp., computational) zero-knowledge proof has a problem-dependent commitment scheme.[4] There are several pieces of evidence pointing to this possibility:

1. A restricted form of a complete problem for statistical zero knowledge has a problem-dependent commitment scheme [MV03].
2. If problem-dependent commitments exist for all problems with statistical zero-knowledge proofs, then problem-dependent commitments exist for all of problems with (general, computational) zero-knowledge proofs [Vad04].
3. Every problem that has (general, computational) zero-knowledge proofs also has inefficient problem-dependent commitments. These commitments are inefficient in the sense that the sender algorithm is not polynomial-time computable [Vad04]. Unfortunately we cannot use these commitments in our protocols in this paper, because our verifier plays the role of the sender.

If the above hypotheses turn out to be true, then our work suggests that we should be able prove that *any* problem that has a zero-knowledge proof has a concurrent zero-knowledge protocol: simply plug the hypothesized problem-dependent commitment scheme into our constructions. (We do not claim this as a theorem because for in this paper, we restrict our attention to problem-dependent commitment schemes that are noninteractive and perfectly binding for simplicity, but the hypotheses mentioned above make no such restriction.)

Outline. Section 2 covers the basic definitions, the various notions of zero knowledge, and the computational problems that we consider in later parts of the paper. In Section 3, we present a modularization of the preamble stage in the Prabhakaran-Rosen-Sahai concurrent zero-knowledge protocol [PRS02, Section 3.1], which we call the *concurrently-extractable commitment scheme*. This commitment scheme is a basic building block for our unconditional concurrent zero-knowledge protocols in Section 4, where the problem-dependent commitments used are of the *perfectly-binding* type, and also in Section 5, where the problem-dependent commitments used are of the *witness-binding* type.

2 Preliminaries

2.1 Basic notations

Let X be a random variable taking values in a finite set T. We write $x \leftarrow X$ to indicate that x is selected according to X. For a finite set S, we write $x \leftarrow S$ to indicate that x is selected uniformly

[4] Actually, the works of [MV03] and [Vad04] refer to problem-dependent commitments where the hiding property holds on YES instances and the binding property on NO instances, which is opposite of what we use. For statistical zero knowledge, this does not matter because the class of problems having statistical zero-knowledge proofs is closed under complement [Oka00]. But for computational zero knowledge, it means that outline presented here might yield a concurrent zero-knowledge *argument* system rather than a proof system.

amongst all the elements of S.

A negligible function, denoted by $\mathrm{neg}(\cdot)$, is a function that grows slower than any inverse polynomial. That is, for all $c \in \mathbb{N}$, $\mathrm{neg}(n) < n^{-c}$ for sufficiently large n.

Let I be a countable set. A *probability ensemble* of a sequence of random variables indexed by I is denoted as $\{X_i\}_{i \in I}$. We say that two ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ are *computationally indistinguishable* (w.r.t. security parameter n) if for every polynomial-sized circuit C, and all sufficiently long $i \in I$, we have that

$$|\Pr[C(X_i, i) = 1] - \Pr[C(Y_i, i) = 1]| < \mathrm{neg}(n).$$

Typically, we set n to be the input or index length. If the above inequality holds for all circuits C (instead of just polynomial-sized ones), the two ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ have a stronger property of being *statistically indistinguishable*. We use the notation $\approx_{\mathrm{c}}$ and $\approx_{\mathrm{s}}$ to denote computational and statistical indistinguishability of ensembles respectively. Note that for computational indistinguishability, we focus on indistinguishability against *nonuniform* circuits, as is standard in the study of zero-knowledge proofs.

For a probabilistic algorithm A, we write $A(x; r)$ to denote the output of A on input x and coin tosses r. $A(x)$, or more precisely $[A(x; r)]_{r \leftarrow \{0,1\}^*}$, is the random variable denoting the output distribution of A on input x with uniformly selected coin tosses r.

For an interactive protocol (A, B), we denote $\langle A, B\rangle(x)$ to be the random variable representing the output of B after interaction with A on common input x. In addition, we denote $\mathrm{view}_B^A(x)$ to be the random variable representing the content of the random tape of B together with the messages received by B from A during the interaction on common input x.

Promise problems. Although all our results apply to languages, for convenience and generality, we state and prove all our results in term of *promise problems*. A promise problem Π is a pair $(\Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}})$ of disjoint sets of strings, corresponding to the YES instances and the NO instances, respectively. This yields the following computational task: Given a string x in $\Pi_{\mathrm{Y}} \cup \Pi_{\mathrm{N}}$, decide whether $x \in \Pi_{\mathrm{Y}}$ or $x \in \Pi_{\mathrm{N}}$. Strings in $\Pi_{\mathrm{Y}} \cup \Pi_{\mathrm{N}}$ are called *instances* of the problem Π.

For a problem $\Pi \in \mathbf{NP}$, we say that R_Π is an $\mathbf{NP}$-relation for Π if $\mathrm{R}_\Pi = \{(x, w) : M(x, w) = 1\}$, where M is an $\mathbf{NP}$-machine for Π.

Statistical difference. The *statistical difference* between two distributions D_1 and D_2 over $\{0,1\}^n$ is defined as $\Delta(D_1, D_2) = \max_{T \subseteq \{0,1\}^n} |\Pr[D_1 \in T] - \Pr[D_2 \in T]|$. For a circuit $X : \{0,1\}^m \to \{0,1\}^n$, the probability distribution induced by X is the output distribution of X when fed a random input, *i.e.,* $X(r)_{r \leftarrow \{0,1\}^m}$. We define the statistical difference between two circuits to be the statistical difference between the distributions induced by the circuits.

2.2 Zero knowledge

We use the standard definition of *zero-knowledge proofs* [GMR89, Gol01], noting the following points.

1. We extend the definition to promise problems $\Pi = (\Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}})$ in a natural way, *i.e.,* conditions previously required for inputs in the language (*e.g.,* completeness and zero knowledge) are now required for all YES instances, and conditions previously required for inputs out of the language (*e.g.,* soundness) are now required for all NO instances.

2. The class **PZK** (*resp.,* **SZK** and **CZK**) denotes the class of promise problems having perfect (*resp.,* statistical and computational) zero-knowledge proofs.

3. A zero-knowledge protocol is called *black-box zero knowledge* if the simulator uses the adversarial verifier only as an oracle. All protocols presented in this paper are black-box zero knowledge.

4. For a problem $\Pi \in \mathbf{NP}$, we say a zero-knowledge protocol has an *efficient prover* if the prover's algorithm can be implemented in probabilistic polynomial-time given any witness w of the instance x.

5. There are two equivalent formulation of zero knowledge (against any general adversarial efficient verifier) for an interactive protocol (P, V):

   (a) For all probabilistic polynomial-time adversary V^*, there exists a probabilistic polynomial-time simulator S such that
   $$\Big\{S(x)\Big\}_{x\in\Pi} \approx \Big\{\langle P, V^*\rangle(x)\Big\}_{x\in\Pi},$$
   where $\langle P, V^*\rangle(x)$ is the random variable representing the output of V^* after interaction with P on common input x.

   (b) For all probabilistic polynomial-time adversary V^*, there exists a probabilistic polynomial-time simulator S such that
   $$\Big\{S(x)\Big\}_{x\in\Pi} \approx \Big\{\mathrm{view}^P_{V^*}(x)\Big\}_{x\in\Pi},$$
   where $\mathrm{view}^P_{V^*}(x)$ is the random variable representing the content of the random tape of V^* together with the messages received by V^* from P during the interaction.

6. A *concurrent interaction* between a prover P and an adversarial verifier V^*, can be thought of as the following interaction. The adversary V^* sends a pair (s, v)—interpreted as sending message v in session s—and the prover P responses by sending the corresponding prover message p for session s. (The prover P behaves independently in all sessions.) This continues until the verifier sends $(\mathtt{end}, \alpha)$, where α is the output. Thus a transcript of a concurrent interaction is of the form $((s_1, v_1), p_1, (s_2, v_2), p_2, ..., (s_t, v_t), p_t, \alpha, \mathtt{end})$. For a commitment scheme (S, R), we can similarly define a concurrent interaction between a receiver R and multiple senders, controlled by a single adversary $\widehat{S}$.

   A protocol is *concurrent zero knowledge* if it remains zero knowledge in a concurrent interaction between the prover P and any probabilistic polynomial-time verifier strategy V^*. This is a much stronger security requirement compared to the standard definition of *stand-alone* zero-knowledge protocols, where the zero-knowledge guarantee is only for a single execution with a single verifier. For an equivalent formulation, we refer the reader to [DNS98].

In addition to the standard notion of zero-knowledge protocols, which remain secure against any efficient adversarial verifier, we consider other notions of zero-knowledge protocols which are guaranteed secure against only a selective subset of adversarial verifiers.[5] We begin with a weak version of zero knowledge that is secure only against the honest verifier.

[5]In the settings where only a selective subset of adversarial verifiers are considered, we will have to require that the simulator's output be indistinguishable to the view of the verifier, not just indistinguishable to the verifier's output. The previously mentioned equivalence in the formulation of zero knowledge works only in the case of general adversaries.

Definition 2.1 (honest-verifier zero knowledge)**.** An interactive proof (P, V) for (promise) problem Π is *perfect (resp., statistical and computational) honest-verifier zero knowledge (HVZK)* if there exists a probabilistic polynomial-time simulator S such that the ensembles $\left\{ \mathrm{view}_V^P(x) \right\}_{x \in \Pi_Y}$ and $\left\{ S(x) \right\}_{x \in \Pi_Y}$ are perfectly (*resp.,* statistically and computationally) indistinguishable.

We also consider interactive protocols where the honest verifier's strategy is to send random coins as its messages.

Definition 2.2 (public-coin proofs [BM88])**.** An interactive proof is *public-coin* if the honest verifier's messages consists of random coin tosses, uniform and independent of the previous messages.

Prabhakaran, Rosen and Sahai [PRS02], in their works on concurrent zero knowledge, showed that adding a $\widetilde{O}(\log n)$-round preamble to a specific form of zero-knowledge protocol (the Hamiltonicity protocol) results in a concurrent zero-knowledge proof system, assuming the existence of a collection of claw-free functions. Alon Rosen, in his PhD thesis, noted that the preamble can be added to a more general form of zero-knowledge protocol, which he informally defines as *challenge-response zero knowledge* [Ros03, Section 4.8.1]. We formalize this definition and call it *committed-verifier zero knowledge.*

Definition 2.3 (committed-verifier zero knowledge)**.** Let $m = (m_1, m_2, \ldots, m_k)$. A *committed verifier V_m* is a deterministic verifier that always sends m_i as its i-th round message.

An interactive proof (P, V) for (promise) problem Π is *perfect (resp., statistical, computational) committed-verifier zero knowledge (CVZK)* if there exists a probabilistic polynomial-time simulator S such that for all committed verifier V_m, the ensembles $\left\{ \mathrm{view}_{V_m}^P(x) \right\}_{x \in \Pi_Y}$ and $\left\{ S(x, m) \right\}_{x \in \Pi_Y}$ are perfectly (*resp.,* statistically, computationally) indistinguishable.

The following lemma shows that the notion of committed-verifier zero knowledge (CVZK) is closely related to honest-verifier zero knowledge (HVZK).

Lemma 2.4. *A promise problem Π has* public-coin *(perfect/statistical/computational) CVZK proofs if and only if it has* public-coin *(perfect/statistical/computational) HVZK proofs.*

Proof. The forward implication is easy since the honest-verifier simulator $S(x)$ can be obtained from the committed-verifier simulator $S'(x, m)$ by choosing a random m.

For the reverse implication, consider the honest-verifier zero-knowledge protocol (P, V) with honest-verifier simulator S. Without loss of generality, we assume that the verifier V always sends the first message and that the verifier's messages in each iteration are of length ℓ. In iteration i, the verifier V sends random coin tosses $m_i \leftarrow \{0, 1\}^\ell$, and prover P responds with $\pi_i = P(x, m_1, \ldots, m_i; r_P)$.

We modify the original protocol (P, V) to the following protocol (P', V'): In iteration i,

$P' \to V'$: Send random coin tosses $m'_i \leftarrow \{0, 1\}^\ell$.

$V' \to P'$: Send random coin tosses $m_i \leftarrow \{0, 1\}^\ell$.

$P' \to V'$: Send $\pi'_i = P(x, m'_1 \oplus m_1, \ldots, m'_i \oplus m_i; r_P)$.

This modified protocol (P', V') preserves the completeness of the original protocol (P, V). Soundness is also preserved because for any fixed m'_i, choosing a random m_i (done by the verifier V') will result in $m'_i \oplus m_i$ being a random message. In addition the number of rounds increases by at most 1, since consecutive prover's messages π'_i and m'_{i+1} can be collapsed into a single round.

Given committed-verifier V'_m, where $m = (m_1, \ldots, m_k)$, our committed-verifier simulator $S'(x, m)$ will run the honest-verifier simulator $S(x)$ to get the transcript $(r_1, \pi_1, \ldots, r_k, \pi_k)$. The output of S' is $(r_1 \oplus m_1, m_1, \pi_1, \ldots, r_k \oplus m_k, m_k, \pi_k)$.

We claim that $S'(x, m)$ is a proper committed-verifier simulator for (P', V'). To prove our claim, define the function $f_m \colon (r_1, \pi_1, \ldots, r_k, \pi_k) \mapsto (r_1 \oplus m_1, m_1, \pi_1, \ldots, r_k \oplus m_k, m_k, \pi_k)$. Note that f_m is polynomial-time computable. Thus, since $S(x)$ and $\text{view}_V^P(x)$ are indistinguishable, so are $S'(x, m) \equiv f_m(S(x))$ and $\text{view}_{V'_m}^P(x) \equiv f_m(\text{view}_V^P(x))$. □

2.3 Computational Problems

We list the computational problems for which we provide unconditional concurrent statistical zero-knowledge proofs.

Quadratic Residuosity. A number x is a *quadratic residue* in the group $\mathbb{Z}_n^* \stackrel{\text{def}}{=} \{a \in \{1, 2, \ldots, n-1\} : \gcd(a, n) = 1\}$ if there exists a $y \in \mathbb{Z}_n^*$ such that $y^2 \equiv x \pmod{n}$. Otherwise, x is a *quadratic nonresidue* in $\mathbb{Z}_n^*$.

The language of quadratic residuosity is defined as follows: Quadratic Residuosity $= \{(x, n) : x \text{ is a quadratic residue in } \mathbb{Z}_n^*\}$. The language Quadratic Nonresiduosity is the complement of Quadratic Residuosity. In the later sections, we consider a variant of quadratic residuosity restricted to *Blum integers*, which we call Blum Quadratic Nonresiduosity. Recall that a Blum integer n is a product of two distinct primes p and q, with $p \equiv q \equiv 3 \pmod{4}$. Formally, we define Blum Quadratic Nonresiduosity = Quadratic Nonresiduosity$\cap \{(x, n) : n \text{ is a Blum integer and } x \in \mathbb{Z}_n^*\}$.

Graph Isomorphism. Two graphs G_0 and G_1 are *isomorphic* if there exists a permutation π of nodes in G_0 that would result in G_1, *i.e.,* $\pi(G_0) = G_1$, in which case we write $G_0 \cong G_1$. The language Graph Isomorphism consists of all pairs of graphs that are isomorphic, that is Graph Isomorphism $= \{(G_0, G_1) : G_0 \cong G_1\}$. The language Graph NonIsomorphism is the complement of Graph Isomorphism.

Statistical Difference. The Statistical Difference problem is the computational task of distinguishing whether the probability distributions induced by the two circuits X and Y are statistically close or far apart. With parameters α and β satisfying $0 \leq \beta < \alpha \leq 1$, the instances of the promise problem SD_β^α are pairs of circuits (X, Y). Pair (X, Y) is a yes instance if $\Delta(X, Y) \geq \alpha$, and a no instance if $\Delta(X, Y) \leq \beta$. The promise problem $\text{SD}_{1/3}^{2/3}$ is **SZK**-complete [SV03]. In Section 4, we provide an unconditional concurrent statistical zero-knowledge proof for a relaxed version of this complete problem, namely $\text{SD}_{1/2}^1$. Currently, we do not know if our result extends to $\text{SD}_{1/3}^{2/3}$, which if true, would then automatically extend to all of **SZK**.

Approximate Lattice Problems. Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. A *lattice* in $\mathbb{R}^m$ is the set of all integral combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$. Denoting the matrix $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_n]$, the lattice generated by basis matrix $\mathbf{B}$ is $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$. The *open ball* of radius r centered at point $\mathbf{a}$ is $\text{B}(\mathbf{a}, r) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{a}\| < r\}$.

Two computational lattice problems that have been widely studied are the approximate versions of the Closest Vector Problem and Shortest Vector Problem. We capture the computational task of approximating the closest lattice point to a given vector by the promise problem

GapCVP$_\gamma$. Instances of this promise problem are pairs $(\mathbf{B}, \mathbf{y}, t)$, where $\mathbf{B}$ is a lattice basis, $\mathbf{y} \in \mathbb{Z}^m$ and $t \in \mathbb{Q}$. The YES instances are $(\mathbf{B}, \mathbf{y}, t)$ such that $\|\mathbf{Bx} - \mathbf{y}\| \leq t$ for some $x \in \mathbb{Z}^n$, while the NO instances are $(\mathbf{B}, \mathbf{y}, t)$ such that $\|\mathbf{Bx} - \mathbf{y}\| > \gamma t$ for all $x \in \mathbb{Z}^n$.

Similarly, we capture the computational task of approximating the shortest lattice point (excluding the origin) to the origin by the promise problem GapSVP$_\gamma$. Instances of this promise problem are pairs $(\mathbf{B}, t)$, where $\mathbf{B}$ is a lattice basis, and $t \in \mathbb{Q}$. The YES instances are $(\mathbf{B}, t)$ such that $\|\mathbf{Bx}\| \leq t$ for some $x \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$, while the NO instances are $(\mathbf{B}, t)$ such that $\|\mathbf{Bx}\| > \gamma t$ for all $x \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$.

For both these problems, a central parameter of interest is γ, the gap between the YES instances and the NO instances. For $\gamma = \Omega(\sqrt{(n/\log n)})$, the problems GapCVP$_\gamma$ and GapSVP$_\gamma$, and their complements, co-GapCVP$_\gamma$ and co-GapSVP$_\gamma$, are known to be in **SZK** [GG00] (unconditionally).[6]

3 Concurrently-Extractable Commitment Scheme

A key component in our concurrent zero-knowledge protocols is a commitment scheme with a *concurrent extractability* property. We call this scheme *concurrently-extractable commitment scheme.* The notion of concurrent extractability informally means that we are able to simulate the concurrent execution of many sessions between an adversarial sender and the honest receiver in a way that allows us to extract the commitments of the sender in every session.

This notion of concurrent extractability is inspired by the rewinding and simulation strategy of the Prabhakaran-Rosen-Sahai (PRS) [PRS02] concurrent zero-knowledge protocol. The PRS protocol essentially consists of two stages, the preamble (first) stage and the main (second) stage [PRS02, Section 3.1]. The concurrent zero knowledge feature of the protocol comes from the preamble stage, in which the verifier is required to commit to the messages that it will use in the main stage. Our goal in designing a concurrently-extractable commitment scheme is to modularize the PRS protocol by abstracting this key feature (preamble stage) that allows for concurrent security.

In our concurrent zero-knowledge protocol, the verifier V plays the role of the sender S and the prover P plays the role of the receiver R. This approach essentially allows us to easily add various other components when designing concurrent zero-knowledge protocols for specific problems, as done in Sections 4 and 5.

We begin with a generic (noninteractive) commitment scheme and show how to transform it into a new scheme with the concurrent extractability property.

Definition 3.1. A *generic (noninteractive) commitment scheme* is a circuit $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$, with n being the security parameter. We define:

1. The *commitment* to a bit b to be $\mathsf{Com}(b; r)$, where $r \leftarrow \{0,1\}^n$ is a uniformly chosen random key.
2. The *decommitment* of c to a bit b to be a pair (b, r) such that $c = \mathsf{Com}(b; r)$.

Note that this definition only refers to the syntax of a commitment scheme and does not impose any security requirements (i.e. hiding and binding).

[6]Goldreich and Goldwasser [GG00] actually showed an honest-verifier **PZK** proof for co-GapCVP$_\gamma$ and co-GapSVP$_\gamma$, but the stated results follow from the closure of **SZK** and also that honest-verifier **SZK** = **SZK** [Oka00, SV03, GV99, GSV98]. Moreover, Micciancio and Vadhan [MV03] gave a direct construction of an efficient-prover **SZK** protocol for both GapCVP$_\gamma$ and GapSVP$_\gamma$.

The concurrently-extractable (interactive) commitment scheme (S, R) consists of two phases, the commit phase and the decommit phase, and is presented below.

Protocol 3.2. Concurrently-extractable commitment scheme (S, R).

Input:

- Parameters k and n (given in unary), where n is the security parameter.
- A generic commitment scheme $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ (given as a circuit).
- A message $m \in \{0,1\}$, to be committed to, given as a private input to sender S.

Commit phase:

$S \to R$: For all $1 \le i \le n$ and $1 \le j \le k$, do the following:

(i) Secret share message m into random shares $m^0_{i,j}$ and $m^1_{i,j}$ such that $m^0_{i,j} \oplus m^1_{i,j} = m$.

(ii) Select random and independent keys $r^0_{i,j}, r^1_{i,j} \leftarrow \{0,1\}^n$.

(iii) Send $c^0_{i,j} = \mathsf{Com}(m^0_{i,j}; r^0_{i,j})$ and $c^1_{i,j} = \mathsf{Com}(m^1_{i,j}; r^1_{i,j})$.

For $j = 1, \ldots, k$, do the following:

$R \to S$: Send $b_{1,j}, \ldots, b_{k,j} \leftarrow \{0,1\}$.

$S \to R$: Send $(m^{b_{1,j}}_{1,j}, r^{b_{1,j}}_{1,j}), \ldots, (m^{b_{k,j}}_{k,j}, r^{b_{k,j}}_{k,j})$, which are decommitments of $c^{b_{1,j}}_{1,j}, \ldots, c^{b_{k,j}}_{k,j}$ respectively.

Decommit phase:

$S \to R$: To decommit to message m, send m together with $\left((m^{1-b_{1,j}}_{i,j}, r^{1-b_{i,j}}_{i,j})\right)_{1 \le i \le n, 1 \le j \le k}$.

R : Accept the decommitment if the following verification passes: Verify that for all i and j, $c^0_{i,j} = \mathsf{Com}(m^0_{i,j}; r^0_{i,j})$, $c^1_{i,j} = \mathsf{Com}(m^1_{i,j}; r^1_{i,j})$, and $m^0_{i,j} \oplus m^1_{i,j} = m$.

Typically, we set the the parameter k to be slightly superlogarithmic, namely $k = \widetilde{O}(\log n)$. In this case, the commit phase has $2k + 1 = \widetilde{O}(\log n)$ rounds.

Next, we show that the above concurrently-extractable commitment scheme has three properties, namely hiding, binding, and concurrent extractability. The hiding and binding properties are inherited from the generic commitments Com. To avoid confusion between the generic commitment scheme and the concurrently-extractable commitment scheme, we call the decommitments to the former, *minor decommitments*, and to the latter, *major decommitments*.

3.1 Hiding Property

In order for the concurrently-extractable commitment scheme to be used as a building block for our concurrent zero-knowledge protocols, we will need to demonstrate that this concurrently-extractable commitment scheme maintains the hiding property. Specifically, we show that if the generic commitment scheme used is hiding, then the message committed to by the sender S will remain hidden even after interacting with any (dishonest) receiver R^*. Let $\text{view}_{R^*}^{S(m)}(\mathsf{Com}, 1^k, 1^n)$ be a random variable representing the messages received by R^* from $S(m)$ in the *commit phase only.*

Lemma 3.3. *Assume the generic commitment scheme* $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ *is statistically hiding, that is the ensembles* $\{\mathsf{Com}(0)\}_{n\in\mathbb{N}}$ *and* $\{\mathsf{Com}(1)\}_{n\in\mathbb{N}}$ *are statistically indistinguishable. Then for any (dishonest) receiver* R^**, for all* $k \le \text{poly}(n)$*, we have that*

$$\left\{ \text{view}_{R^*}^{S(0)}(\mathsf{Com}, 1^k, 1^n) \right\}_{n\in\mathbb{N}} \approx_{\text{s}} \left\{ \text{view}_{R^*}^{S(1)}(\mathsf{Com}, 1^k, 1^n)) \right\}_{n\in\mathbb{N}}.$$

Proof. Let $\preceq$ be any total ordering on the set $T = \{(i,j)\}_{1\le i\le n, 1\le j\le k} \cup \{(0,0)\}$ with the property that $(0,0)$ and (n,k) are the least and largest elements respectively.

Define the following hybrid senders $\{S_{\alpha,\beta}\}_{(\alpha,\beta)\in T}$: $S_{\alpha,\beta}$ acts like S with the only exception that during the first message of the commit phase, it chooses independent random shares $m^0_{i,j}$ and $m^1_{i,j}$ of 0 when $(i,j) \succ (\alpha,\beta)$, and independent random shares $m^0_{i,j}$ and $m^1_{i,j}$ of 1 when $(i,j) \preceq (\alpha,\beta)$. Observe that $S_{\alpha,\beta}$ does not have any private input, and that $S_{0,0} = S(0)$ and $S_{n,k} = S(1)$.

Note that since $k \le \text{poly}(n)$, the size of set T is $nk + 1 \le \text{poly}(n)$. Using a hybrid argument, it is sufficient to show that for any (α,β) and its immediate successor (γ,δ), it is the case that $\text{view}_{R^*}^{S_{\alpha,\beta}} \approx_{\text{s}} \text{view}_{R^*}^{S_{\gamma,\delta}}$, dropping the common inputs $(\mathsf{Com}, 1^k, 1^n)$ for simplicity of notation.

The only difference between $S_{\alpha,\beta}$ and $S_{\gamma,\delta}$ is that the former chooses independent random shares $m^0_{\alpha,\beta}$ and $m^1_{\alpha,\beta}$ of 0, while the latter chooses independent random shares $m^0_{\alpha,\beta}$ and $m^1_{\alpha,\beta}$ of 1. Since the generic commitment scheme Com is statistically hiding, and both $S_{\alpha,\beta}$ and $S_{\gamma,\delta}$ only reveal one of their random shares, we have that $\text{view}_{R^*}^{S_{\alpha,\beta}} \approx_{\text{s}} \text{view}_{R^*}^{S_{\gamma,\delta}}$. □

Committing to multi-bit messages. The concurrently-extractable commitment scheme, presented as Protocol 3.2, is for a single-bit message; to commit to a ℓ-bit message, we repeat the above protocol ℓ times in parallel. It is important to note that even if we do so, the multi-bit message will remain hidden before the decommit phase. This fact can be proven by extending Lemma 3.3 with a standard hybrid argument on the multi-bit message. Specifically, to show indistinguishability of committed ℓ-bit messages M and M', we construct $\ell + 1$ different hybrid senders with the first and last having M and M' as their private inputs respectively, and each neighboring hybrid senders having private inputs differing by only a single bit.

Furthermore, when we commit to multi-bit messages, we can decommit in multiple steps, one for each committed bit. This is because the decommit phase for each bit of the message is independent of the other decommit phases for the rest of the message.

3.2 Binding Property

We demonstrate the binding property of the concurrently-extractable commitment scheme via a reduction analysis. Informally stated, we show that if a dishonest sender S^* can major-decommit to two different messages, then we can break the binding property of the generic commitment scheme Com (which is impossible if Com is binding). In fact, we will prove (and use) a stronger statement, which, instead of requiring two inconsistent major decommitments, only requires *one* major decommitment together with minor decommitments for one of the k^2 sharings of the message $m = m^0_{i,j} \oplus m^1_{i,j}$. (Recall that a major decommitment consists of minor decommitments for all k^2 sharings.) The following lemma states this precisely.

Lemma 3.4. *For all dishonest sender S^*, let $C = \mathrm{view}_R^{S^*}(\mathsf{Com}, 1^k, 1^n)$ be the transcript of the commit phase. For some i and j, let $d^0_{i,j} = (m^0_{i,j}, r^0_{i,j})$ and $d^1_{i,j} = (m^1_{i,j}, r^1_{i,j})$ be minor decommitments of $c^0_{i,j}$ and $c^1_{i,j}$ contained in C. Specifically, $c^0_{i,j} = \mathsf{Com}(m^0_{i,j}; r^0_{i,j})$ and $c^1_{i,j} = \mathsf{Com}(m^1_{i,j}; r^1_{i,j})$, and both $c^0_{i,j}$ and $c^1_{i,j}$ are part of the sender's first message in C.*

Let $m = m^0_{i,j} \oplus m^1_{i,j}$. If there exist a major decommitment D of C to a different message $\tilde{m} \neq m$, then from the values of D, C, $d^0_{i,j}$ and $d^1_{i,j}$, we can efficiently compute an r and r' such that $\mathsf{Com}(0; r) = \mathsf{Com}(1; r')$.

Proof. Since D is a valid major decommitment for C, from these values, we obtain minor decommitments $(\tilde{m}^0_{i,j}, \tilde{r}^0_{i,j})$ and $(\tilde{m}^1_{i,j}, \tilde{r}^1_{i,j})$ of $c^0_{i,j}$ and $c^1_{i,j}$ respectively, for all i and j. In addition, $\tilde{m} = \tilde{m}^0_{i,j} \oplus \tilde{m}^1_{i,j}$, for all i and j.

However, it is also the case that $d^0_{i,j} = (m^0_{i,j}, r^0_{i,j})$ and $d^1_{i,j} = (m^1_{i,j}, r^1_{i,j})$ are minor decommitments of $c^0_{i,j}$ and $c^1_{i,j}$, for some i and j. Since $m^0_{i,j} \oplus m^1_{i,j} = m \neq \tilde{m} = \tilde{m}^0_{i,j} \oplus \tilde{m}^1_{i,j}$, either $m^0_{i,j} \neq \tilde{m}^0_{i,j}$ or $m^1_{i,j} \neq \tilde{m}^1_{i,j}$. In both cases, we have two different minor decommitments to the same commitment (either $c^0_{i,j}$ or $c^1_{i,j}$). $\square$

3.3 Concurrent Extractability Property

Recall that the prover P and adversarial verifier V^* (in our concurrent zero-knowledge protocols) will play the role of the receiver R and concurrent adversarial sender, denoted as $\widehat{S}$, respectively. Therefore, we will need to exhibit a simulation strategy for R against any adversarial sender $\widehat{S}$ concurrently interacting with it. Doing so by itself is trivial since R only sends random coin tosses and thus does not leak knowledge. However, in order to use the concurrently-extractable commitment scheme as a building block for our concurrent zero-knowledge protocols, a key property required is the ability of this simulator to also determine the initial committed message of $\widehat{S}$ in all sessions. We call this the *concurrent extractability* property, a notion that we will later formalize. First, we make certain simplifying assumptions on $\widehat{S}$.

Assumptions on the concurrent adversary. Without loss of generality, we impose certain restrictions on the concurrent adversarial sender $\widehat{S}$. These restrictions will aid in simplifying our proofs of concurrent security (in Sections 4 and 5).

- $\widehat{S}$ is a deterministic algorithm. As in the usually done in the analysis of zero-knowledge protocols, the adversary's random coin tosses can be fixed in advanced.
- $\widehat{S}$ executes *exactly* Q concurrent sessions, for some parameter Q that is typically bounded by $\mathrm{poly}(n)$. This assumption can be made because we can easily upper bound the number of sessions by the running time of $\widehat{S}$. And if $\widehat{S}$ executes less than Q sessions, we can easily modify $\widehat{S}$ to send dummy messages in the sessions that it does not execute, making it execute exactly Q sessions.
- For each sender round $j = 1, \ldots, k$ in the commit phase, $\widehat{S}$ always either sends valid decommitments or the message `error`. This is because a decommitment (b, r) to a commitment c can be easily checked for validity by testing if $\mathsf{Com}(b; r) = c$.

- $\widehat{S}$ only interacts with R in the commit phase. This is because our simulation only deals with that phase.
- $\widehat{S}$ commits to an ℓ-bit message. Although our concurrently-extractable commitment scheme handles only single-bit messages, as stated previously, we can repeat the protocol in parallel to get a multi-bit commitment scheme. Since our simulator handles concurrent interaction, parallel repetition is just a special case and will not pose a problem. Later, it will be more convenient to think of $\widehat{S}$ as committing to an ℓ-bit message per session, rather than ℓ-senders committing to a single-bit message each. Hence, we will include ℓ as an additional parameter to the simulator.

Simulator for the concurrently-extractable commitment scheme. We denote the simulator for the commit phase of Protocol 3.2 as $\mathsf{CEC\text{-}Sim}$, with the term CEC being an acronym for concurrently-extractable commitments. Simulator $\mathsf{CEC\text{-}Sim}$ has access to oracle $\widehat{S}$ and is given the following inputs:

- Generic commitment schemes $\mathcal{COM} = (\mathsf{Com}_1, \mathsf{Com}_2, \ldots, \mathsf{Com}_Q)$, where $\mathsf{Com}_s \colon \{0,1\}\times\{0,1\}^n \to \{0,1\}^n$ is the commitment scheme used for session s and is given as a circuit.
- Parameters ℓ, k, n and Q, all given in unary. Note that n is the security parameter and Q is the number of sessions executed by $\widehat{S}$.

The concurrently extractability property of the simulator will restrict the way in which $\mathsf{CEC\text{-}Sim}$ queries $\widehat{S}$. Before providing a formal definition, we define the following notions of *valid commit phase transcript* and *compatibility.*

Valid commit phase. For a transcript T of the commit phase interaction between S and R, let $T[s]$ denote the messages in session s. $T[s]$ is a *valid commit phase transcript* if there exist a major decommitment D such that $R(T[s], D) = \texttt{accept}$. In particular, this implies that all of the sender's minor decommitments in the commit phase $T[s]$ are valid (i.e. the sender never sends $\texttt{error}$), which is the only property of valid commit phase transcripts that is used in Lemma 3.6 below.

Compatibility. Message $M = (m, m^0_{i,j}, r^0_{i,j}, m^1_{i,j}, r^1_{i,j})$ is *compatible* with $T[s]$ if

(i) $m = m^0_{i,j} \oplus m^1_{i,j}$, and

(ii) $\mathsf{Com}_s(m^0_{i,j}; r^0_{i,j}) = c^0_{i,j}$ and $\mathsf{Com}_s(m^1_{i,j}; r^1_{i,j}) = c^1_{i,j}$, with $c^0_{i,j}[s]$ and $c^1_{i,j}[s]$ being part of the first message in $T[s]$.

Observe that M contains a potential committed message m of the sender in session s, together with minor decommitments of shares of m. By Lemma 3.4, it is impossible for the sender to major-decommit to a message different from m without breaking the binding property of Com. Thus we call m the *extracted message.*

Definition 3.5 (simulator with concurrent extraction)**.** Simulator $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$ has the *concurrent extraction* property if for every query T it makes to $\widehat{S}$, it also provides (on a separate output tape) an array of messages $(M_1, M_2, \ldots, M_Q)$ with the following property:

For every session $s \in \{1, 2, \ldots, Q\}$, if $T[s]$ is a valid commit phase transcript, then M_s is compatible with $T[s]$.

A simulator that has the concurrently extractable property is also called a *concurrently-extractable simulator*.

Using the simulation and rewinding techniques in [PRS02], we obtain a concurrently-extractable simulator (for Protocol 3.2) that runs in probabilistic polynomial time. Recall that $\langle R, \widehat{S} \rangle$ denotes the output of $\widehat{S}$ after concurrently interacting with R, and that $\widehat{S}$ can be a computationally unbounded adversary.

Lemma 3.6 (implicit in [PRS02])**.** *There exists a probabilistic polynomial-time concurrently-extractable simulator* $\mathsf{CEC\text{-}Sim}$ *such that for all generic commitment schemes* $\mathcal{COM}$ *and all concurrent adversarial sender* $\widehat{S}$*, for settings of parameter* $\ell = \mathrm{poly}(n)$*,* $k = \widetilde{O}(\log n)$*, and* $Q = \mathrm{poly}(n)$*, we have the ensembles* $\left\{ \mathsf{CEC\text{-}Sim}^{\widehat{S}}(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q) \right\}_{n \in \mathbb{N}}$ *and* $\left\{ \langle R, \widehat{S} \rangle (\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q) \right\}_{n \in \mathbb{N}}$ *are statistically indistinguishable.*

Remark 3.7. Simulator $\mathsf{CEC\text{-}Sim}$ has essentially the same rewinding strategy as the simulator in [PRS02], which we call the PRS simulator. Our definition of concurrently-extractable simulators is motivated by the following fact:

> Whenever the PRS simulator completes a simulation of a session s (that is, $T[s]$ is a complete commit phase), with high probability, it solves the session by providing decommitments to $c^0_{i,j}$ and $c^1_{i,j}$, for some i and j.

In the case when the PRS simulator fails to solve the session (and this happens with negligible probability), our simulator $\mathsf{CEC\text{-}Sim}$ will halt and not query $\widehat{S}$.

Another difference between the PRS simulator and $\mathsf{CEC\text{-}Sim}$ is that the former is only required to provide compatible messages for the most recently completed commit phase, whereas $\mathsf{CEC\text{-}Sim}$ is required to provide compatible messages for *every* completed commit phase. However, since the PRS simulator has already solved the previous sessions in prior simulation steps, it actually has the ability to provide compatible messages for every completed commit phase (though not explicitly stated in [PRS02]).

4 Unconditional Concurrent Zero-Knowledge Proofs for Problems with Perfectly-Binding Commitments

In this section, we demonstrate a generic technique of transforming stand-alone public-coin zero-knowledge protocols for certain **coNP** problems into concurrent zero-knowledge protocols. In doing so, we construct the first known unconditional concurrent zero-knowledge protocols for non-trivial problems like Quadratic Nonresiduosity, Graph NonIsomorphism, the statistical difference problem $\mathrm{SD}^1_{1/2}$ and approximate lattice problems.

The main tool used in the transformation is a *problem-dependent commitment scheme*, formally defined in Definition 4.1. Later in Section 5, we demonstrated a modified transformation that works for certain **NP** problems.

4.1 Perfectly-Binding Problem-Dependent Commitments

In order to prevent the adversarial verifier from deviating widely from the original protocol specification, the previous constructions of concurrent zero-knowledge protocols require the verifier to commit to certain messages in advance [RK99, KP01, PRS02]. While these commitments can be constructed from one-way functions [Nao91, HILL99], proving the existence of one-way functions remains a major open problem in complexity theory.

To achieve concurrent security without relying on unproven assumptions, we observe that the standard verifier's commitments used in [PRS02] can be replaced by *problem-dependent commitments* [IOS97] (*cf.,* [MV03]). A problem-dependent commitment, roughly speaking, is a commitment protocol that takes the problem instance x as an additional input, is binding on the YES instances ($x \in \Pi_{\mathrm{Y}}$), and is hiding on the NO instances ($x \in \Pi_{\mathrm{N}}$). Standard commitments, by contrast, are required to always be both hiding and binding regardless of the problem instance.

Because the hiding and binding properties of problem-dependent commitments depend on the problem instance, we can construct problem-dependent commitments that both perfectly binding (on the YES instances) and statistically hiding (on the NO instances).[7] We give a simplified, non-interactive definition of problem-dependent commitments that suffices for our applications in this section.

Definition 4.1 (noninteractive problem-dependent commitment)**.** A promise problem $\Pi = (\Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}})$ has a *perfectly-binding problem-dependent commitment* if there exists a polynomial-time algorithm $\mathsf{PD\text{-}Com}$ such that the following holds.

1. Algorithm $\mathsf{PD\text{-}Com}$ takes as input the problem instance x, a bit b, and a random key r, and produces a commitment $c = \mathsf{PD\text{-}Com}_x(b; r)$. The running time of $\mathsf{PD\text{-}Com}$ is bounded by a polynomial in $|x|$, and without loss of generality we can assume that $|c| = |r| = \mathrm{poly}(|x|)$.

2. (perfectly binding on YES instances) For all $x \in \Pi_{\mathrm{Y}}$, the distributions $\mathsf{PD\text{-}Com}_x(0)$ and $\mathsf{PD\text{-}Com}_x(1)$ have disjoint supports. That is, there does not exist strings r and r' such that $\mathsf{PD\text{-}Com}_x(0; r) = \mathsf{PD\text{-}Com}_x(1; r')$.

3. (statistically hiding on NO instances) For all $x \in \Pi_{\mathrm{N}}$, the commitments to 0 and 1 are statistically indistinguishable. In other words, the distributions $\mathsf{PD\text{-}Com}_x(0)$ and $\mathsf{PD\text{-}Com}_x(1)$ are statistically indistinguishable (w.r.t. $|x|$, the length of the instance).

The commitment c can be decommitted to by sending the committed bit b and random key r. Since both parties have access to the problem instance x, this decommitment can be verified by checking that $c = \mathsf{PD\text{-}Com}_x(b; r)$.

Consider the statistical difference problem $\mathrm{SD}^1_{1/2}$. This problem involves deciding whether two distributions are statistically close or are disjoint. The perfectly-binding problem-dependent commitment for $\mathrm{SD}^1_{1/2}$, due to Micciancio and Vadhan [MV03], is essentially as follows: Let $x = (X_0, X_1)$ be an instance of $\mathrm{SD}^1_{1/2}$. To commit to bit b, select a random r and send $X_b(r)$ as the commitment. To decommit, send b and r. Because the distributions induced by X_0 and X_1 are disjoint for the YES instances, the commitment is binding on those instances. In the case of the NO instances, the statistical distance between the two distributions are at most 1/2. This value can be further reduced to $2^{-|x|}$, a negligible value, by an XOR Lemma [SV03].

Observe that any problem that has a perfectly-binding problem-dependent commitment in the sense of Definition 4.1 is in **coNP**; the witness for the NO instances is (r, r') such that $\mathsf{PD\text{-}Com}_x(0; r) = \mathsf{PD\text{-}Com}_x(1; r')$. In fact, there is a tight characterization between perfectly-binding problem-dependent commitments and the statistical difference problem $\mathrm{SD}^1_{1/2} \in \mathbf{coNP}$.

Lemma 4.2 ([MV03])**.** *Promise problem* Π *has a perfectly-binding problem-dependent commitment if and only if* Π *reduces to* $\mathrm{SD}^1_{1/2}$*.*

[7] By contrast, standard commitments *cannot* be both statistically binding and statistically hiding.

To accommodate problems outside **coNP**, it is possible to relax the definition of problem-dependent commitments (Definition 4.1) to allow for interaction, and also to allow for computational binding on the YES instances (because the commitment is being used by polynomial-time bounded verifier). We consider both these relaxations in Section 5.1 (where we construct a modified problem-dependent commitment scheme for certain **NP** problems). Nevertheless, the *efficiency of the sender* in the commitment scheme is crucial because our protocols will require the verifier (acting as the sender) to do the problem-dependent commitments. Thus, although Vadhan [Vad04] constructed problem-dependent commitments for all of zero knowledge, we cannot use those commitments because they utilize an inefficient sender.

As we will later show, being binding on the YES instances and hiding on the NO instances is exactly the property of the verifier's commitment that we need in our unconditional concurrent zero-knowledge protocol. This property of our commitment scheme is very similar to the definition of positively transparent and negatively opaque commitments in [IOS97, Definitions 2.1 & 2.2], and to that used in an earlier work of [BMO90] which implicitly uses a form of problem-dependent commitments to prove an unconditional constant-round **PZK** protocol for GRAPH ISOMORPHISM.

Some other works, like [MV03, Vad04] and [IOS97, Section 4.1], that have employed the use of problem-dependent commitments defined these commitments to have exactly the opposite property, namely hiding on the YES instances and binding on the NO instances. This is because their protocols require the prover to do the problem-dependent commitments, whereas our protocols and that of [BMO90] and [IOS97, Section 4.2] require the verifier to do the problem-dependent commitments.

4.2 Main Results

Before presenting the our unconditional concurrent zero-knowledge protocol, we state our main results for this section.

Theorem 4.3. *If promise problem Π has a public-coin CVZK proof system (P_0, V_0) and also a perfectly-binding problem-dependent commitment, then Π has a proof system (P, V) with the following properties:*

1. *Zero-knowledge guarantee is preserved. That is, if (P_0, V_0) is statistical (resp., computational) zero knowledge, then (P, V) is concurrent statistical (resp., computational) zero knowledge.*
2. *Prover P is black-box simulatable in* strict *polynomial time.*
3. *The round complexity of (P, V) increases only by an additive factor of $\widetilde{O}(\log n)$, with n being the security parameter, compared to the original protocol (P_0, V_0).*
4. *The completeness of (P, V) is exactly the same as that of (P_0, V_0), while the soundness error increases by only a negligible additive term (as a function of n).*
5. *The prover strategy P can be implemented in probabilistic polynomial-time with oracle access to P_0. In particular, if P_0 is efficient, so is P.*

Note that the requirement of Π having public-coin CVZK is equivalent to having public-coin HVZK (by Lemma 2.4). We provide a full proof of Theorem 4.3 in Sections 4.3 and 4.4. The following corollary of Theorem 4.3 follows from the fact that $\mathrm{SD}^1_{1/2}$ exactly characterizes problems with perfectly-binding problem-dependent commitments (Lemma 4.2), and has public-coin statistical zero-knowledge proofs [Oka00, SV03].

Corollary 4.4. *If promise problem* Π *has perfectly-binding problem-dependent commitment, then* Π *has a concurrent statistical zero-knowledge proof.*

Furthermore, any problem that reduces to $\mathrm{SD}^1_{1/2}$ has concurrent statistical zero-knowledge proofs, by Lemma 4.2 and the above corollary. Several natural problems that fit into this category are Quadratic Nonresiduosity, Graph NonIsomorphism, and the approximate lattice problems, co-GapCVP$_\gamma$ and co-GapSVP$_\gamma$ (these reductions are implicit in [GMR89, GMW91, GG00, SV03, MV03]).

Corollary 4.5. *The following problems have concurrent statistical zero-knowledge proofs:*

- *The statistical difference problem* $\mathrm{SD}^1_{1/2}$.
- *The languages* Quadratic Nonresiduosity *and* Graph NonIsomorphism.
- *The lattice problems* co-GapCVP$_\gamma$ *and* co-GapSVP$_\gamma$, *for* $\gamma = \Omega(\sqrt{(n/\log n)})$.

The above corollary does not guarantee a polynomial-time prover strategy (with auxiliary input) nor round efficiency. The reason is that the public-coin honest-verifier zero-knowledge proof systems known for these problems do not have a polynomial-time prover nor a subpolynomial number of rounds. For Blum Quadratic Nonresiduosity, however, we can start with the noninteractive statistical zero-knowledge proof[8] of [BDMP91], whose prover is polynomial time (given the factorization of the modulus), and obtain the following:

Corollary 4.6. *The language* Blum Quadratic Nonresiduosity *has a concurrent statistical zero-knowledge proof systems with* $\widetilde{O}(\log n)$ *rounds and a prover that can be implemented in polynomial time given the factorization of the input modulus.*

We note that we do not expect to obtain efficient provers for Graph NonIsomorphism or $\mathrm{SD}^1_{1/2}$, since these problems are not known to be in **NP** (or **MA**), which is a prerequisite for an efficient-prover proof system. However, Quadratic Nonresiduosity is in **NP** (the factorization of the input is a witness), as are co-GapCVP$_\gamma$ and co-GapSVP$_\gamma$ for larger approximation factors $\gamma = \Omega(\sqrt{n})$ [AR04], so we could hope to obtain an efficient prover. The bottleneck is finding *public-coin* honest-verifier zero-knowledge proofs with a polynomial-time prover for these problems. (Such private-coin proof systems are known [GMR89, GG00, AR04].)

4.3 Our Concurrent Zero-Knowledge Protocol

A high-level description of our unconditional concurrent zero-knowledge protocol is as follows: We begin with a public-coin CVZK protocol. We make it concurrent zero knowledge by forcing the verifier to commit in advance to its (public-coin) messages in the CVZK protocol using the concurrently-extractable commitment scheme (Protocol 3.2). Finally, we replace the verifier's standard commitments with problem-dependent commitments, hence not requiring any complexity assumption.

Now, let us formally describe our concurrent zero-knowledge protocol. Let (P_0, V_0) be a public-coin CVZK proof system for Π with $q(|x|)$ rounds on common input x. Denote the messages sent by V_0 in the protocol as $m = (m(1), \ldots, m(q))$, and let $\ell \stackrel{\text{def}}{=} |m|$ be the verifier-to-prover communication complexity. Let $\mathsf{PD\text{-}Com}_x \colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$, where $n = \mathrm{poly}(|x|)$, be a

[8]Noninteractive zero knowledge implies (in fact is equivalent to) 2-round honest-verifier zero knowledge since the honest verifier just sends the common random string in the first round, and the prover sends the single-message proof in the second round.

perfectly-binding problem-dependent commitment for Π. The full description of our concurrent zero-knowledge protocol (P, V) is next.

Protocol 4.7. Our unconditional concurrent zero-knowledge protocol (P, V) for problem Π with perfectly-binding problem-dependent commitments.

Input: Instance x of Π.

Preamble stage (using problem-dependent commitments)

V selects a random message $m \leftarrow \{0,1\}^\ell$, and runs the concurrent commitment scheme (Protocol 3.2) ℓ times in parallel, with V as the sender and P as the receiver. The inputs are the message m, commitment scheme $\mathsf{PD\text{-}Com}_x \colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$, and parameters $k = \widetilde{O}(\log|x|)$ and $n = \mathrm{poly}(|x|)$. The full description of the protocol follows.

$V \to P$: Send the message `"start session"`.

$V \to P$: Select a random message $m = (m(1), m(2), \ldots, m(q(|x|)) \leftarrow \{0,1\}^\ell$. For all $1 \le i, j \le k$, do the following:

(i) Secret share message m into random shares $m^0_{i,j}$ and $m^1_{i,j}$ such that $m^0_{i,j} \oplus m^1_{i,j} = m$.

(ii) Select random and independent keys $r^0_{i,j}, r^1_{i,j} \leftarrow \{0,1\}^n$.

(iii) Send $c^0_{i,j} = \mathsf{PD\text{-}Com}_x(m^0_{i,j}; r^0_{i,j})$ and $c^1_{i,j} = \mathsf{PD\text{-}Com}_x(m^1_{i,j}; r^1_{i,j})$.

For $j = 1, \ldots, k$, do the following:

$P \to V$: Send $b_{1,j}, \ldots, b_{k,j} \leftarrow \{0,1\}^k$.

$V \to P$: Send $(m^{b_{1,j}}_{1,j}, r^{b_{1,j}}_{1,j}), \ldots, (m^{b_{k,j}}_{k,j}, r^{b_{k,j}}_{k,j})$, which are decommitments of $c^{b_{1,j}}_{1,j}, \ldots, c^{b_{k,j}}_{k,j}$ respectively.

Main stage (stand-alone zero-knowledge protocol)

$V \to P$: Send the message `"start main stage"`.

P : Select randomness $r_{P_0} \leftarrow \{0,1\}^*$ for the original prover P_0.

For $t = 1, \ldots, q(|x|)$, do the following:

$V \to P$: Send $m(t)$ and decommit to all the secret shares of $m(t)$, other than those decommitted in the preamble stage. Specifically, decommit to $\{m(t)^{1-b_{i,j}}_{i,j}\}^k_{i,j=1}$.

$P \to V$: Verify that the decommitments are all valid and that $m(t) = m(t)^{1-b_{i,j}}_{i,j} \oplus m(t)^{b_{i,j}}_{i,j}$, for all $i, j \in [1, k]$. If verification fails, halt and abort. Otherwise, answer as the original prover P_0 would, that is, send $\pi_t = P_0(x, m(1), \ldots, m(t); r_{P_0})$.

Verifier V accepts if the original verifier V_0 accepts on $(m(1), \pi_1, m(2), \pi_2, \ldots, m(q), \pi_q)$.

From the Protocol 4.7, we can easily derive the prover efficiency, round complexity and completeness claims of Theorem 4.3. In addition, the soundness follows from the hiding property of the concurrently-extractable commitment scheme (Lemma 3.3). This is because a cheating prover will not know the committed messages of the verifier until the verifier decommits to all secret shares of $m(t)$ (in round t of the main stage). We summarize our results in the following lemma, deferring the proof of concurrent zero knowledge claim to Section 4.4.

Lemma 4.8. *The interactive protocol (P, V), given in Protocol 4.7, has the following properties (with the numbering consistent with Theorem 4.3):*

3. *The round complexity of (P, V) increases only by an additive factor of $\widetilde{O}(\log n)$, with n being the security parameter, compared to the original protocol (P_0, V_0).*

4. *The completeness of (P, V) is exactly the same as that of (P_0, V_0), while the soundness error increases by only a negligible additive term (as a function of n).*

5. *The prover strategy P can be implemented in probabilistic polynomial-time with oracle access to P_0. In particular, if P_0 is efficient, so is P.*

4.4 Our Simulator

The goal of this subsection is to design a simulator for Protocol 4.7. In the next lemma, we summarize the results contained in this subsection.

Lemma 4.9. *The interactive protocol (P, V), given in Protocol 4.7, has the following properties (with the numbering consistent with Theorem 4.3):*

1. *Zero-knowledge guarantee is preserved. That is, if (P_0, V_0) is statistical (resp., computational) committed-verifier zero knowledge, then (P, V) is concurrent statistical (resp., computational) zero knowledge.*

2. *Prover P is black-box simulatable in* strict *polynomial time.*

Observe that Lemma 4.9 combined with Lemma 4.8 from the previous subsection, yield Theorem 4.3. Let us get back to the task of designing a simulator for Protocol 4.7. Observe that the prover's strategy can be broken into two parts, P_{pre} and P_{main},[9] denoting the preamble and the main stage, respectively. Both P_{pre} and P_{main} use independent randomness. The simulation procedure for our concurrent zero-knowledge protocol (Protocol 4.7) is broken into three main steps, which we outline now and detail in the following subsections.

1. First, we analyze the concurrent interaction of P and V^* in the context of the concurrently-extractable commitment scheme (Protocol 3.2). To do so, we define a new adversarial sender $\widehat{S}$ that takes V^* and P_{main} as oracles and only returns the preamble messages of V^*. The preamble stage prover P_{pre} acts as the honest receiver. By Lemma 3.6, we can simulate the output of $\widehat{S}$ (after interaction with P_{pre}), while having the additional property of being able to extract the commitments.

   By virtue of the way we define $\widehat{S}$, its output after concurrently interacting with P_{pre} is equivalent to the output of V^* after concurrently interacting with P. Nevertheless, this simulation is inefficient because $\widehat{S}$ uses an oracle for P_{main}.

[9] The difference between P_{main}, the prover strategy in the main stage, and P_0, the original CVZK prover, is that P_{main} checks for the validity of the decommitments before responding as P_0 would. For the remainder of this section, we will focus on P_{main} instead of P_0.

2. Since we can extract the commitments, we are able to determined the verifier's main stage messages in advance. Hence, we can replace the adaptive queries to P_{main} by a single query made to a new oracle, called $\mathcal{O}_P$, at the start of each main stage.

3. However, $\mathcal{O}_P$ is still not an efficiently implementable oracle. In the final step, we replace oracle $\mathcal{O}_P$ with a committed-verifier zero knowledge (CVZK) simulator S_{CVZK} to obtain an efficient simulation strategy.

We will show that the simulation in Step 1 is indistinguishable from $\langle P, V^* \rangle$, and that in Steps 2 and 3, the output of the simulation in that step is indistinguishable to the previous. We formalize these ideas below.

4.4.1 First simulation procedure

Before presenting our first simulator, we would need to consider different kinds of transcripts and the operation of the concurrent adversarial sender $\widehat{S}$.

Preamble and full transcripts. A *preamble transcript*, denoted as $\mathcal{T}_{\text{pre}}$, is a (partial) transcript of the concurrent interaction between P and V^*, with the main stage messages removed. Hence, it only consists of preamble stage messages and is compatible with the transcript of the commit phase interaction between R and $\widehat{S}$.

A *full transcript*, denoted as $\mathcal{T}_{\text{full}}$, is a (partial) transcript of the concurrent interaction between P and V^*, with all the messages intact. Note that preamble and full transcripts are allowed to capture only a partial interaction between P and V^*.

We say a (preamble or full) transcript is *complete* if it captures the entire concurrent interaction between P and V^*. In the case of complete preamble transcripts, the main stage messages are removed, but all preamble stage interaction are present.

It is important to note that the adversarial verifier V^* operates on full transcripts, whereas the adversarial sender $\widehat{S}$, defined next, operates on preamble transcripts.

Concurrent adversarial sender $\widehat{S}$. Let $\mathcal{F}(\cdot;\cdot)$ be a *truly random function*, that is over a random choice of $r_{\mathcal{F}}$, $\mathcal{F}(\cdot; r_{\mathcal{F}})$ is a random function. We are required to simulate the concurrent interaction between P and V^*. We analyze this interaction in the context of the concurrently extractable commitment scheme (Protocol 3.2). In doing so, we view the preamble stage prover P_{pre} as the honest receiver R, and define a new adversarial sender $\widehat{S}$ that takes V^* and P_{main} as oracles and only returns the preamble messages of V^*. The inputs to $\widehat{S}$ are transcript T and randomness $r_{\mathcal{F}}$ (the random key for the truly random function $\mathcal{F}$).

We give a high-level explanation on the role of $\widehat{S}$. It is designed to mimic V^*, but only outputs its preamble messages. To simulate the main stage messages in some session $s \in \{1, 2, \ldots, Q\}$, $\widehat{S}$ uses oracle P_{main} with prover randomness $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$, where $v_1[s]$ is the first preamble message in session s. The inputs to $\widehat{S}$ are a preamble transcript $\mathcal{T}_{\text{pre}}$ and randomness $r_{\mathcal{F}}$ (for the truly random function $\mathcal{F}$). We will first need to convert $\mathcal{T}_{\text{pre}}$ into a full transcript $\mathcal{T}_{\text{full}}$ with main stage messages. To do so, for every session s, we use oracle P_{main} (with prover randomness $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$) to obtain the prover main stage messages, and use V^* to determine the verifier's main stage messages as well as the scheduling of the messages. A formal definition of $\widehat{S}$ is given next.

$\widehat{S}^{P_{\text{main}},V^*}(\mathcal{T}_{\text{pre}}; r_{\mathcal{F}})$:

1. Let $\mathcal{T}_{\text{pre}} = ((s_1, v_1), p_1, (s_2, v_2), p_2, \ldots, (s_t, v_t), p_t)$.
2. Initialize $\mathcal{T}_{\text{full}} = ()$ and $j = 1$.
3. Query oracle V^* on $\mathcal{T}_{\text{full}}$ to obtain $(s, v) = V^*(\mathcal{T}_{\text{full}})$. Depending on the value of v, s, and t, do the following.

   Case 1: v is a preamble stage message and $j = t + 1$. In this case, send (s, v).

   Case 2: $(s, v) = (\texttt{end}, \alpha)$ and $j = t + 1$. In this case, output α and halt.

   Case 3: v is a preamble stage message, $j \leq t$, and $(s, v) = (s_j, v_j)$. In this case, update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s_j, v_j), p_j)$ and $j = j + 1$. Repeat Step 3.

   Case 4: v is a main stage message. Proceed as follows.

   (a) Let $T[s]$ denote all the messages of session s in $\mathcal{T}_{\text{full}}$, and let $v_1[s]$ be the verifier's first preamble message in $T[s]$.

   (b) Set the main stage prover's randomness $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$.

   (c) Query oracle P_{main} to obtain $p = P_{\text{main}}(x, T[s] \circ v; r_{P_0})$.

   (d) Update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s, v), p)$, and repeat Step 3.

   Case 5: Otherwise, halt and output $\texttt{fail}$, indicating that $\mathcal{T}_{\text{pre}}$ was not a valid preamble transcript.

First simulator. We are now ready to present our first simulation procedure. Observe that $\widehat{S}$ (with $r_{\mathcal{F}}$ fixed) can be thought of as a deterministic oracle taking preamble transcripts $\mathcal{T}_{\text{pre}}$ as queries. In order to simplify notation, we write $\widehat{S}[r_{\mathcal{F}}](\cdot)$ to represent the deterministic oracle $\widehat{S}^{P_{\text{main}},V^*}(\cdot; r_{\mathcal{F}})$.

$\mathsf{Sim\text{-}One}^{P_{\text{main}},V^*}(x; r_{\mathcal{F}})$:

1. Let Q be the bound on the number of concurrent sessions executed by V^*. Set $\mathcal{COM} = \{\mathsf{Com}_1, \mathsf{Com}_2, \ldots, \mathsf{Com}_Q\}$, where $\mathsf{Com}_s = \mathsf{PD\text{-}Com}_x$ (the problem-dependent commitment) for all $s \in \{1, 2, \ldots, Q\}$.
2. Output $\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}]}(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$.

For the rest of the section, we simplify notations by representing $\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}]}(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$ with a shorter notation $\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}]}$, and avoid repeating common inputs $(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$.

Lemma 4.10. *If the number of concurrent sessions Q is bounded by $\text{poly}(|x|)$, then the output of the first simulator $\left\{\mathsf{Sim\text{-}One}^{P_{\text{main}},V^*}(x)\right\}_{x\in\Pi_Y}$ (over random coin tosses $r_{\mathcal{F}} \leftarrow \{0,1\}^*$) is statistically indistinguishable from $\left\{\langle P, V^*\rangle(x)\right\}_{x\in\Pi_Y}$.*

Proof. With settings of parameters $k = \omega(\log|x|)$, $\ell = \text{poly}(|x|)$, $n = \text{poly}(|x|)$ and $Q = \text{poly}(|x|)$, by Lemma 3.6, we have that for every string $r_{\mathcal{F}}$,

$$\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}]} \approx_{\text{s}} \langle P_{\text{pre}}, \widehat{S}[r_{\mathcal{F}}]\rangle.$$

For every session s, the main stage prover's randomness for that session is given by $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$. If $r_{\mathcal{F}}$ is chosen uniformly at random, the value of r_{P_0} will be independent and uniform for each session s. Hence, for uniformly chosen coin tosses $r_{\mathcal{F}}$, the output of $\widehat{S}[r_{\mathcal{F}}]$ after concurrently interacting with P_{pre} is equivalent to the output of V^* after concurrently interacting with P. Stated formally,

$$\left[\langle P_{\text{pre}}, \widehat{S}[r_{\mathcal{F}}]\rangle\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} \equiv \langle P, V^* \rangle(x).$$

Finally we have,

$$\begin{aligned}
\left[\mathsf{Sim\text{-}One}^{P_{\text{main}},V^*}(x; r_{\mathcal{F}})\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} &\stackrel{\text{def}}{=} \left[\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}]}\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} \\
&\approx_{\text{s}} \left[\langle P_{\text{pre}}, \widehat{S}[r_{\mathcal{F}}]\rangle\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} \\
&\equiv \langle P, V^* \rangle(x),
\end{aligned}$$

completing our proof. □

4.4.2 Second simulation procedure

The inefficiency in the first simulator Sim-One stems from the implementation of the adversarial sender $\widehat{S}^{P_{\text{main}},V^*}(T; r_{\mathcal{F}})$. The adversarial sender uses P_{main} as well as a truly random function $\mathcal{F}$ whose key $r_{\mathcal{F}}$ is exponentially long. To overcome that inefficiency, we will mimic the implementation of $\widehat{S}$ without using P_{main} or $\mathcal{F}$, but by employing the use of a probabilistic oracle $\mathcal{O}_P$ defined below.

Oracle $\mathcal{O}_P$: On query $m = (m(1), \ldots, m(q))$, do the following.

1. Select randomness $r_{P_0} \leftarrow \{0,1\}^*$.
2. Output all of the original prover P_0's responses to m, namely $(\pi_t = P_0(x, m(1), \ldots, m(t); r_{P_0}))_{1 \le t \le q}$.

Each new query to oracle $\mathcal{O}_P$ results in choosing *independent* random coin tosses r_{P_0}.

Note that the only use of $r_{\mathcal{F}}$ by $\widehat{S}$ is in generating the main stage prover's randomness $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$. Since $\mathcal{F}$ is a truly random function, the randomness r_{P_0} can be generated on-the-fly without the need for choosing the very long random string $r_{\mathcal{F}}$.

Furthermore, we can determine the V^*'s main stage messages in advance because CEC-Sim is a concurrently-extractable simulator. Whenever it queries $\widehat{S}$ on $\mathcal{T}_{\text{pre}}$, it also provides messages M_s for each complete session $T[s]$. Note that each M_s contains an m that should be V^*'s main stage message in session s. Hence, we can query $\mathcal{O}_P$ on m to obtain $(\pi_1, \ldots, \pi_q)$ instead of doing multiple queries to P_{main}. In order to give consistent main stage prover responses when we rewind the simulation, we record these responses in an array, denoted as Prover-Msg. We formalize these ideas in our second simulator.

$\mathsf{Sim\text{-}Two}^{\mathcal{O}_P,V^*}(x)$**:**

1. *(Similar to* Sim-One*)* Let Q be the bound on the number of concurrent sessions executed by V^*. Set $\mathcal{COM} = \{\mathsf{Com}_1, \mathsf{Com}_2, \ldots, \mathsf{Com}_Q\}$, where $\mathsf{Com}_s = \mathsf{PD\text{-}Com}_x$ (the problem-dependent commitment) for all $s \in \{1, 2, \ldots, Q\}$.
2. Initialize the array of main stage prover responses $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] = \perp$ for all pairs $(v_1[s], s)$.

3. Without using oracle P_{main} and randomness $r_{\mathcal{F}}$, run $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$ until $\mathsf{CEC\text{-}Sim}$ queries $\widehat{S}$ on some $\mathcal{T}_{\text{pre}}$. When this happens, do the following:
   (a) Freeze the execution of $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$.
   (b) Let $T[s]$ denote all the preamble stage messages of session s in $\mathcal{T}_{\text{pre}}$, and let $v_1[s]$ be the verifier's first (preamble) message in $T[s]$. If $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] \neq \perp$, then let $(\pi_1, \dots, \pi_q) = \mathsf{Prover\text{-}Msg}[(v_1[s], s)]$, and proceed to Step 3d.
   (c) For every s such that $T[s]$ is a valid commit phase transcript, simulator $\mathsf{CEC\text{-}Sim}$ provides an extracted message m. Using m, query $\mathcal{O}_P(m)$ to obtain the prover responses $(\pi_1, \dots, \pi_q)$. Update $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] = (\pi_1, \dots, \pi_q)$.
   (d) Instead of querying oracle P_{main} directly, check that the decommitments provided by $\widehat{S}$ in session s are all valid. If they are all valid, use π_t to answer the query to P_{main} for the t-th round prover response in the main stage of session s.
   (e) Continue the execution of $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$ as in Step 3.

By virtue of the way we defined $\mathsf{Sim\text{-}Two}$, it acts exactly like $\mathsf{Sim\text{-}One}$.

Lemma 4.11. *For all $x \in \Pi_{\text{Y}}$, we have that*

$$\left[\mathsf{Sim\text{-}One}^{P_{\text{main}}, V^*}(x; r_{\mathcal{F}})\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} \equiv \mathsf{Sim\text{-}Two}^{\mathcal{O}_P, V^*}(x).$$

Furthermore, $\mathsf{Sim\text{-}Two}$ *is a probabilistic polynomial-time algorithm with oracle access to $\mathcal{O}_P$ and V^*.*

Proof. The fact that $\mathsf{Sim\text{-}Two}$ is a probabilistic polynomial-time algorithm is easy to check. We will show that $\mathsf{Sim\text{-}Two}$ perfectly mimics the execution of $\mathsf{Sim\text{-}One}$. The only times when the execution of $\mathsf{Sim\text{-}Two}$ could potentially differ from $\mathsf{Sim\text{-}One}$ is when we freeze the execution of $\mathsf{CEC\text{-}Sim}$ (in $\mathsf{Sim\text{-}Two}$) because $\widehat{S}^{P_{\text{main}}, V^*}(T; r_{\mathcal{F}})$ requires an answer from oracle P_{main} to simulate the prover's main stage messages in some session s.

In that case, $T[s]$ is a valid commit phase transcript, and $\mathsf{CEC\text{-}Sim}$ provides an extracted message $m = (m(1), \dots, m(q))$ for that session s. By the binding property of the concurrently-extractable commitments (Lemma 3.4) and by the fact that $\mathsf{PD\text{-}Com}_x$ is perfectly binding (for $x \in \Pi_{\text{Y}}$), the message $m = (m(1), \dots, m(q))$ must be the V^* main stage messages in session s.[10]

Given that we know the verifier's main stage messages to be $m = (m(1), \dots, m(q))$, and that $\mathcal{F}$ is a truly random function, the following two random processes result in equivalent output distributions: (i) querying P_{main} multiple times to obtain $(\pi_t = P_0(x, m(1), \dots, m(t); r_{P_0}))_{1 \le t \le q}$, with $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$ and uniformly chosen random coins $r_{\mathcal{F}} \leftarrow \{0,1\}^*$, and (ii) querying $\mathcal{O}_P$ once to obtain $(\pi_t = P_0(x, m(1), \dots, m(t); r_{P_0}))_{1 \le t \le q}$, with uniformly chosen random coins $r_{P_0} \leftarrow \{0,1\}^*$. In addition, in process (ii), we keep a history of the main stage prover responses that we have used in the array $\mathsf{Prover\text{-}Msg}$ to avoid giving a different set of prover responses for the same pair $(v_1[s], s)$. This is important because in process (i), the value of $(v_1[s], s)$ determines the prover's randomness, and hence the prover responses.

Consequently, in the case when $\mathsf{Sim\text{-}Two}$ queries oracle P_{main}, we can use oracle $\mathcal{O}_P$ to supply identically distributed answers. This completes our proof. □

[10] If the verifier halts during step t in the main stage of session s, the prior messages must be $(m(1), \dots, m(t-1))$.

4.4.3 Replacing the oracle $\mathcal{O}_P$

We now proceed to replace the oracle $\mathcal{O}_P$ with a committed-verifier zero knowledge simulator S_{CVZK} for Π. To do so, we define the following probabilistic oracle $\mathcal{O}_{\text{CVZK}}$.

Oracle $\mathcal{O}_{\text{CVZK}}$: On query m, do the following.

1. Select randomness $r_S \leftarrow \{0,1\}^*$.
2. Let the output of $S_{\text{CVZK}}(x, m; r_S)$ be $(m(1), \pi_1, \ldots, m(q), \pi_q)$, where $m = (m(1), \ldots, m(q))$.
3. Output only the simulated prover responses $(\pi_1, \ldots, \pi_q)$.

By the definition of CVZK, we have that for all $x \in \Pi_{\text{Y}}$ and messages m, the distribution of the transcript $\langle P_0, V_m\rangle(x)$ is indistinguishable from $S(x, m)$. Hence, the two oracles $\mathcal{O}_P$ and $\mathcal{O}_{\text{CVZK}}$ produce indistinguishable answers when asked the same query. The final step is to show that the simulation $\mathsf{Sim\text{-}Two}^{\mathcal{O}_P, V^*}(x)$ is indistinguishable from $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\text{CVZK}}, V^*}(x)$. To do so, we employ the following lemma, whose proof follows from a standard hybrid argument.

Lemma 4.12. *Let n denote the security parameter, and let $\mathcal{A}$ and $\mathcal{B}$ be probabilistic oracles. If $[\mathcal{A}(q; r_A)]_{r_A \leftarrow \{0,1\}^*} \approx_{\text{s}} [\mathcal{B}(q; r_B)]_{r_B \leftarrow \{0,1\}^*}$ for all query q, then for all $D^{\mathcal{O}}$ that queries its oracle $\mathcal{O}$ at most* $\text{poly}(n)$ *number of times, we have $D^{\mathcal{A}} \approx_{\text{s}} D^{\mathcal{B}}$.*

When the two oracles produce only computationally indistinguishable answers, the above statement will hold as long as D is an efficient algorithm and one of the two oracles, say $\mathcal{A}$, is efficient. Naturally, the resulting distributions $D^{\mathcal{A}}$ and $D^{\mathcal{B}}$ will only be computationally indistinguishable.

Since $\mathsf{Sim\text{-}Two}$ runs in polynomial time (Lemma 4.11), it queries its oracles at most polynomially many times. Hence we have that $\left\{\mathsf{Sim\text{-}Two}^{\mathcal{O}_P, V^*}(x)\right\}_{x \in \Pi_{\text{Y}}} \approx \left\{\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\text{CVZK}}, V^*}(x)\right\}_{x \in \Pi_{\text{Y}}}$, with the strength of the indistinguishability (statistical or computational) depending on the type of CVZK protocol we started from.

4.4.4 Final simulator

Our final simulator for Protocol 4.7 is $\mathsf{Sim\text{-}Three}^{V^*}(x) \stackrel{\text{def}}{=} \mathsf{Sim\text{-}Two}^{\mathcal{O}_{\text{CVZK}}, V^*}(x)$. By Lemmas 4.10 and 4.11, we have that $\left\{\mathsf{Sim\text{-}Three}^{V^*}(x)\right\}_{x \in \Pi_{\text{Y}}} \approx \left\{\langle P, V^*\rangle(x)\right\}_{x \in \Pi_{\text{Y}}}$. Since S_{CVZK} is a probabilistic polynomial-time algorithm, oracle $\mathcal{O}_{\text{CVZK}}$ is efficiently computable. Therefore, $\mathsf{Sim\text{-}Three}$ is a black-box simulator that runs in *strict* polynomial time. This proves Lemma 4.9, thus completing our proof of Theorem 4.3.

5 Unconditional Concurrent Zero-Knowledge Proofs for Problems with Witness-Binding Commitments

Here we extend the techniques in Section 4 to obtain unconditional concurrent statistical zero-knowledge proofs for certain problems like QUADRATIC RESIDUOSITY and GRAPH ISOMORPHISM.

5.1 Witness-Binding Problem-Dependent Commitments

Based on the techniques used in Section 4, the first natural step towards constructing concurrent zero-knowledge protocols would be to construct problem-dependent commitments. Consider the naive commitment scheme for GRAPH ISOMORPHISM specified as follows: Let (G_0, G_1) be an instance of the problem. To commit to bit b, send a random isomorphic copy of G_b.

This commitment is perfectly hiding on the YES instances (when $G_0 \cong G_1$) and perfectly binding on the NO instances (when $G_0 \not\cong G_1$). However, this is exactly the opposite of what we require in a problem-dependent commitment (see Definition 4.1). In fact, every problem satisfying Definition 4.1 is in **coNP**, but GRAPH ISOMORPHISM is not known to be in **coNP**.

To overcome this apparent difficulty, we will make use of an additional setup phase to do problem-dependent commitments. Let S be the sender, the party that commits, and R be the receiver, the party that receives the commitment. Consider the following commitment scheme, which is similar to that used in [BMO90].

Protocol 5.1. Witness-binding problem-dependent commitment scheme for GRAPH ISOMORPHISM (implicit in [BMO90]).

To commit to bit b using problem instance (G_0, G_1), proceed as follows.

Index generation stage

$R \to S$: Let H_1 be a random isomorphic copy of G_0, and send H_1. That is, $H_1 = \sigma(G_0)$ for a random permutation σ of the vertices of G_0. In addition, both parties set $H_0 = G_0$.

Commitment stage

$S \to R$: To commit to bit b, send F, a random isomorphic copy of H_b.

Decommitment stage

$S \to R$: To decommit, send b together with the isomorphism between H_b and F.

Verification stage

After the decommitment stage, the receiver R proves that H_1, sent in the index generation stage, is isomorphic to G_0 by sending the isomorphism σ between G_0 and H_1.

As we will later prove, the above commitment scheme is perfectly hiding on every instance (in particular the NO instances) if H_1 is generated correctly, that is if $H_1 \cong G_0$. On the YES instances, the scheme is "computationally binding" in that breaking the scheme is as hard as finding an **NP**-

witness (an isomorphism between G_0 and G_1). This scheme can be generalized to many other **NP** languages, and we do so by abstracting the key properties of Protocol 5.1.

Definition 5.2. A *witness-binding problem-dependent commitment scheme* for a promise problem $\Pi \in \mathbf{NP}$ is a collection of five polynomial-time algorithms ($\mathsf{Generate}, \mathsf{Verify}, \mathsf{WB\text{-}Com}, \mathsf{Simulate}, \mathsf{Extract}$) as described below.

$\mathsf{Generate}(x; r_S)$ takes an instance x of Π and randomness r_S, and produces a pair (z, π) (where z is an index, and π is "proof" that z is a good index).

$\mathsf{WB\text{-}Com}_{x,z}(b; r_C)$ takes an instance x of Π, an index z, a bit b and coin tosses r_C, and outputs a commitment c.

$\mathsf{Verify}(x, z, \pi)$ takes an instance x of Π, an index z and a proof π, and outputs `accept` or `reject`.

$\mathsf{Simulate}(x; r_E)$ takes an instance x of Π and coin tosses r_E, and outputs some index z.

$\mathsf{Extract}(x, d, d', r_E)$ takes a string x, decommitments d and d', and coin tosses of the $\mathsf{Simulate}$ algorithm r_E, and outputs a string w (which is hopefully an **NP**-witness for x).

We require the following properties.

1. *(Honest setup passes verification)* For every $x \in \Pi_{\mathrm{Y}}$ and r_S, if $(z, \pi) = \mathsf{Generate}(x; r_S)$, then $\mathsf{Verify}(x, z, \pi) = \texttt{accept}$.

2. *(Statistically hiding on* NO *instances if index string is good)* For every $x \in \Pi_{\mathrm{N}}$ and z, if there exists π such that $\mathsf{Verify}(x, z, \pi) = \texttt{accept}$, then the distributions $\mathsf{WB\text{-}Com}_{x,z}(0)$ and $\mathsf{WB\text{-}Com}_{x,z}(1)$ are statistically indistinguishable (w.r.t. $|x|$).

3. *(Simulated index string looks real on* YES *instances)* For every $x \in \Pi_{\mathrm{Y}}$, taking $z \leftarrow \mathsf{Simulate}(x)$ is distributed identically to the first component of $\mathsf{Generate}(x)$.

4. *(Non-binding commitments yield an* **NP***-witness)* For every $x \in \Pi_{\mathrm{Y}}$, if $z = \mathsf{Simulate}(x; r_E)$ and $\mathsf{WB\text{-}Com}_{x,z}(0, r_C) = \mathsf{WB\text{-}Com}_{x,z}(1, r'_C)$, then $w = \mathsf{Extract}(x, (0, r_C), (1, r'_C), r_E)$ is an **NP**-witness for x.

We observe that Protocol 5.1, the problem-dependent commitment scheme for Graph Isomorphism, is indeed witness-binding.

Lemma 5.3. *The language* Graph Isomorphism *has witness-binding problem-dependent commitments.*

Proof. The $\mathsf{Generate}$, $\mathsf{WB\text{-}Com}$ and Verify algorithms follow directly from Protocol 5.1. We first prove that Property 2 of Definition 5.2 is satisfied, namely $\mathsf{WB\text{-}Com}$ is statistically hiding (on every instance) if the index string is good. Let S_n be the symmetric permutation group. If there exists a π such that $\mathsf{Verify}((G_0, G_1), H_1, \pi)$ accepts, *i.e.,* H_1 is isomorphic to G_0, then observe that

$$\begin{aligned}
\left[\mathsf{WB\text{-}Com}_{((G_0,G_1),H_1)}(0;\pi_0)\right]_{\pi_0 \leftarrow S_n} &\equiv [\pi_0(G_0)]_{\pi_0 \leftarrow S_n} \\
&\equiv [\pi_1(H_1)]_{\pi_1 \leftarrow S_n} \quad \text{(since } H_1 \text{ is isomorphic to } G_0\text{)} \\
&\equiv \left[\mathsf{WB\text{-}Com}_{((G_0,G_1),H_1)}(1;\pi_1)\right]_{\pi_1 \leftarrow S_n}
\end{aligned}$$

To prove that non-binding commitments yield an **NP**-witness (Property 4), we define the $\mathsf{Simulate}$ and $\mathsf{Extract}$ algorithms. The $\mathsf{Simulate}$ algorithm takes (G_0, G_1) and a permutation γ, and outputs $H_1 = \gamma(G_1)$. If the commitment is not binding, then there exists permutations π_0 and π_1 such that $\pi_0(H_0) = F = \pi_1(H_1)$ (using the same notations as in Protocol 5.1). Note that $H_0 = G_0$ and we have chosen $H_1 = \gamma(G_1)$. The $\mathsf{Extract}$ algorithm takes (G_0, G_1), the decommitments π_0 and π_1, and coins γ used by the $\mathsf{Simulate}$ algorithm, to produce an isomorphism $\widetilde{\gamma} = \pi_0^{-1}\pi_1\gamma$ between G_0 and G_1, specifically $G_0 = \widetilde{\gamma}(G_1)$.

The other properties, namely Properties 1 and 3, can be easily check. □

Lemma 5.4. *The language* Quadratic Residuosity *has witness-binding problem-dependent commitments.*

Proof. We present a witness-binding commitment scheme for Quadratic Residuosity.

> To commit to bit b using problem instance (x, n), proceed as follows.
>
> **Index generation stage**
>
> $R \to S$: Select a random $\alpha \leftarrow \mathbb{Z}_n^*$, and send $\beta = \alpha^2 \pmod n$.
>
> **Commitment stage**
>
> $S \to R$: To commit to bit b, select a random $r \leftarrow \mathbb{Z}_n^*$, and send $c = x\beta^b r^2 \pmod n$.
>
> **Decommitment stage**
>
> $S \to R$: To decommit, send (b, r).
>
> **Verification stage**
>
> After the decommitment stage, the receiver R proves that β, sent in the index generation stage, is a quadratic residue in $\mathbb{Z}_n^*$ by sending the square root α. The sender S checks that $\beta \equiv \alpha^2 \pmod n$.

Similar to the proof of Lemma 5.3, the $\mathsf{Generate}$, $\mathsf{WB\text{-}Com}$ and Verify algorithms follow directly from the above protocol. We first prove that Property 2 of Definition 5.2 is satisfied, namely $\mathsf{WB\text{-}Com}$ is statistically hiding (on every instance) if the index string is good. If there exists an $\alpha \in \mathbb{Z}_n^*$ such that $\mathsf{Verify}((x, n), \beta, \alpha)$ accepts, *i.e.,* $\beta \equiv \alpha^2 \pmod n$ for some $\alpha \in \mathbb{Z}_n^*$, then observe that

$$\begin{aligned}
[\mathsf{WB\text{-}Com}_{x,\beta}(0; r)]_{r \leftarrow \mathbb{Z}_n^*} &\equiv \left[xr^2 \pmod n\right]_{r \leftarrow \mathbb{Z}_n^*} \\
&\equiv \left[x(\alpha\tilde{r})^2 \pmod n\right]_{\tilde{r} \leftarrow \mathbb{Z}_n^*} \quad \text{(multiplication by } \alpha \text{ is a permutation in } \mathbb{Z}_n^*) \\
&\equiv \left[x\beta\tilde{r}^2 \pmod n\right]_{\tilde{r} \leftarrow \mathbb{Z}_n^*} \\
&\equiv [\mathsf{WB\text{-}Com}_{x,\beta}(1; \tilde{r})]_{\tilde{r} \leftarrow \mathbb{Z}_n^*}
\end{aligned}$$

To prove that non-binding commitments yield an **NP**-witness (Property 4), we define the $\mathsf{Simulate}$ and $\mathsf{Extract}$ algorithms. The $\mathsf{Simulate}$ algorithm takes (x, n) and an integer $\alpha \in \mathbb{Z}_n^*$, and outputs $\beta \equiv \alpha^2 x^{-1} \pmod n$. If the commitment is not binding, then there exists integers r and $\tilde{r}$ such that

$$xr^2 \pmod n = \mathsf{WB\text{-}Com}_{x,\beta}(0; r) = \mathsf{WB\text{-}Com}_{x,\beta}(1; \tilde{r}) = x\beta\tilde{r}^2 \pmod n.$$

When we use the simulated $\beta \equiv \alpha^2 x^{-1} \pmod{n}$, observe that $x \equiv x\beta\tilde{r}^2 r^{-2} \equiv \alpha^2\tilde{r}^2 r^{-2} \pmod{n}$, and hence a square root of x in $\mathbb{Z}_n^*$ is $\alpha\tilde{r}r^{-1} \pmod{n}$.

With this in mind, the Extract algorithm takes (x, n), the decommitments $(0, r)$ and $(1, \tilde{r})$, and coins used by the Simulate algorithm α, to produce a square root of x, namely $\alpha\tilde{r}r^{-1} \pmod{n}$. This proves that non-binding commitments yield an **NP**-witness, satisfying Property 4 of Definition 5.2.

The other properties, namely Properties 1 and 3, can be easily check. □

5.2 Witness-Completable CVZK

In Section 4, we transformed public-coin CVZK protocols into concurrent zero-knowledge protocols using perfectly-binding problem-dependent commitment. To obtain unconditional concurrent zero-knowledge proofs using only witness-binding commitments, we require the underlying stand-alone protocol to have a stronger property of being *witness-completable* CVZK. The additional witness-completable property, informally stated, gives our simulator the ability to complete the simulation even when the verifier sends a message different from its committed one, if we provide our simulator with a valid witness. For simplicity, we restrict our definition to the special case of 3-round protocols.

Definition 5.5 (witness-completable CVZK)**.** An 3-round interactive proof (P, V) for (promise) problem Π is statistical *witness-completable committed-verifier zero knowledge (wCVZK)* if the following three conditions hold.

1. P is an *efficient* prover, that is, the prover algorithm can be implemented in probabilistic polynomial-time given a valid witness w of $x \in \Pi_{\mathrm{Y}}$.

2. For all valid witness w, $(P(w, r), V)$ is a statistical CVZK protocol. Specifically, letting S_{CVZK} denote the CVZK simulator, we have that for all $x \in \Pi_{\mathrm{Y}}$ and any corresponding **NP**-relation R_Π, and all committed verifier V_m, the ensembles $\left\{ \mathrm{view}_{V_m}^{P(w,r)}(x) \right\}_{(x,w)\in \mathrm{R}_\Pi}$ and $\left\{ S_{\mathrm{CVZK}}(x, m) \right\}_{x\in\Pi_{\mathrm{Y}}}$ are statistically indistinguishable.

3. There exists a probabilistic polynomial-time witness-completable simulator S_{wc} taking as input the instance x, witness w, committed message m, actual verifier message m' and randomness of the CVZK simulator $\tilde{r}$, and outputs the second simulated message of the prover (on verifier message m').

   Formally, we have that for all $x \in \Pi_{\mathrm{Y}}$, any valid witness w of x, every message m, and every circuit D,
$$\left| \begin{array}{l} \Pr_r[D^{P_2(x,w,\cdot,r)}(\mathrm{view}_{V_m}^{P(w,r)}(x)) = 1] - \\ \Pr_{\tilde{r}}[D^{S_{\mathrm{wc}}(x,w,m,\cdot,\tilde{r})}(S_{\mathrm{CVZK}}(x, m, \tilde{r})) = 1] \end{array} \right| = \mathrm{neg}(n),$$
   where $P_2(w, \cdot, r)$ is the second message of the prover on witness w and randomness r, given as a oracle-function of the verifier's message. The notation $\mathrm{view}_{V_m}^{P(w,r)}(x)$ denotes the transcript of a single execution of the prover P, using witness w and randomness r, with a committed-verifier V_m.

We define 3-round *computational* wCVZK in an analogous fashion, with the requirements relaxed to allow for $(P(w, r), V)$ to be computational CVZK, and distinguishing circuit D to be polynomial size.

The main additional security property of wCVZK protocols over their CVZK counterpart is the requirement that $\text{view}_{V_m}^{P(w,r)}(x)$ and $S_{\text{CVZK}}(x, m, \tilde{r})$ remain indistinguishable even when the distinguisher D has oracle access to $P_2(x, w, \cdot, r)$ and $S_{\text{wc}}(x, w, m, \cdot, \tilde{r})$ respectively. Next, we show that both GRAPH ISOMORPHISM and QUADRATIC RESIDUOSITY have public-coin perfect wCVZK proofs.

Lemma 5.6. *The language* GRAPH ISOMORPHISM *has an efficient prover, public-coin, perfect wCVZK proof system.*

Proof. Consider the standard interactive proof for GRAPH ISOMORPHISM [GMW91].

> **Input:** Graphs (G_0, G_1) given as common input to both P and V, and permutation ϕ given as a private input to P. The witness is a permutation ϕ such that $G_0 = \phi(G_1)$.
>
> $P \to V$: Select a random permutation π, and send $H = \pi(G_0)$.
>
> $V \to P$: Send a random bit $b \leftarrow \{0, 1\}$.
>
> $P \to V$: If $b = 0$, send $\psi = \pi$. Else if $b = 1$, send $\psi = \pi \circ \phi$.
>
> V: Verifier V accepts if only if $H = \psi(G_b)$.

Clearly, P has an efficient prover strategy. The proof that (P, V) is a perfect zero-knowledge protocol [GMW91] can be easily modified to show that it is also committed-verifier perfect zero knowledge. Specifically, simulator S_{CVZK} on inputs (G_0, G_1) and b, selects a random permutation π and outputs $(H = \pi(G_b), b, \pi)$.

Our witness-completable simulator S_{wc} takes instance (G_0, G_1), witness ϕ, message b, alternate message $\bar{b}$ and CVZK simulator randomness π, and outputs π if $\bar{b} = b$, $\pi \circ \phi$ if $\bar{b} \neq b = 0$, or $\pi \circ \phi^{-1}$ if $\bar{b} \neq b = 1$. We can check that both S_{CVZK} and S_{wc} satisfy the requirements of being witness-completable CVZK (see Definition 5.5). □

Lemma 5.7. *The language* QUADRATIC RESIDUOSITY *has an efficient prover, public-coin, perfect wCVZK proof system.*

Proof. Consider the standard interactive proof for QUADRATIC RESIDUOSITY [GMR89].

> **Input:** Integers (x, n) given as common input to both P and V, and integer y given as a private input to P. The witness is a $y \in \mathbb{Z}_n^*$ such that $y^2 \equiv x \pmod{x}$.
>
> $P \to V$: Select a random integer $r \leftarrow \mathbb{Z}_n^*$, and send $\alpha = r^2 \pmod{n}$.
>
> $V \to P$: Send a random bit $b \leftarrow \{0, 1\}$.
>
> $P \to V$: If $b = 0$, send $\beta = r$. Else if $b = 1$, send $\beta = yr \pmod{n}$.
>
> V: Verifier V accepts if only if $\beta^2 \equiv \alpha x^b \pmod{n}$.

Clearly, P has an efficient prover strategy. The proof that (P, V) is a perfect zero-knowledge protocol [GMR89] can be easily modified to show that it is also committed-verifier perfect zero knowledge. Specifically, simulator S_{CVZK} on inputs (x, n) and b, selects a integer $r \leftarrow \mathbb{Z}_n^*$ and outputs $(r^2 x^{-b} \pmod{n}, b, r)$.

Our witness-completable simulator S_{wc} takes instance (x, n), witness y, message b, alternate message $\bar{b}$ and CVZK simulator randomness r, and outputs r if $\bar{b} = b$, $ry \pmod{n}$ if $\bar{b} \neq b = 0$, or $ry^{-1} \pmod{n}$ if $\bar{b} \neq b = 1$. We can check that both S_{CVZK} and S_{wc} satisfy the requirements of being witness-completable CVZK (see Definition 5.5). □

5.3 Main Results

Our main result for this section can be summarized by the following theorem.

Theorem 5.8. *If promise problem Π has a 3-round, public-coin, wCVZK proof system (P_0, V_0) and also a witness-binding problem-dependent commitment, then Π has a concurrent zero-knowledge proof system (P, V) with the following properties:*

1. *Zero-knowledge guarantee is preserved. That is, if (P_0, V_0) is statistical (resp., computational) zero knowledge, then (P, V) is concurrent statistical (resp., computational) zero knowledge.*
2. *Prover P is black-box simulatable in* expected *polynomial time.*
3. *The round complexity of (P, V) increases only by an additive factor of $\widetilde{O}(\log n)$, with n being the security parameter, compared to the original protocol (P_0, V_0).*
4. *The completeness of (P, V) is exactly the same as that of (P_0, V_0), while the soundness error increases by only a negligible additive term (as a function of n).*
5. *The prover strategy P can be implemented in probabilistic polynomial-time with oracle access to P_0. In particular, if P_0 is efficient, so is P.*

We provide a full proof of Theorem 5.8 in Sections 5.4 and 5.5. The following theorem follows from Theorem 5.8, together with Lemmas 5.3, 5.4, 5.6 and 5.7.

Theorem 5.9. *Both languages* Graph Isomorphism *and* Quadratic Residuosity *have concurrent statistical zero-knowledge proof systems with $\widetilde{O}(\log n)$ rounds and efficient provers. The simulator for both protocols runs in* expected *polynomial-time.*

Although the stand-alone wCVZK protocol for both Graph Isomorphism and Quadratic Residuosity has high soundness error of 1/2, we can reduce the error to negligible (while maintaining the round complexity) in the resulting concurrent zero-knowledge protocol via parallel repetition. This parallelized protocol remains concurrent zero knowledge because parallel repetition is a special case of concurrent interaction.

Finally, note that the round complexity of $\widetilde{O}(\log n)$ for the concurrent zero-knowledge protocols of both Graph Isomorphism and Quadratic Residuosity is essentially optimal for black-box simulation [CKPR03].

5.4 Our Modified Concurrent zero-knowledge protocol

We modify Protocol 4.7, our concurrent zero-knowledge protocol in Section 4.3, to accommodate witness-binding commitments. Let (P_0, V_0) be a 3-round, public-coin *witness-completable* CVZK proof system for Π. On common input x, let m be the single message sent by V_0 in the protocol, and let $\ell \stackrel{\text{def}}{=} |m|$ be the verifier's communication complexity. Let $\mathsf{WB\text{-}Com}_{x,z}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$, where $n = \mathrm{poly}(|x|)$, be a witness-binding problem-dependent commitment for Π. The full description of our concurrent zero-knowledge protocol (P, V) is next.

Protocol 5.10. Our unconditional concurrent zero-knowledge protocol for problem Π with witness-binding commitments.

Input: Instance x of Π.

$V \to P$: Send the message `"start session"`.

Index generation stage

$P \to V$: Choose a random r_S, and let $(z, \pi) = \mathsf{Generate}(x; r_S)$. Send z.

Preamble stage (using witness-binding commitments)

V selects a random message $m \leftarrow \{0,1\}^\ell$, and runs the concurrent commitment scheme (Protocol 3.2) ℓ times in parallel, with V as the sender and P as the receiver. The inputs are the message m, commitment scheme $\mathsf{WB\text{-}Com}_{x,z}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$, and parameters $k = \widetilde{O}(\log|x|)$ and $n = \mathrm{poly}(|x|)$.

Main stage (stand-alone zero-knowledge protocol)

$V \to P$: Send the message `"start main stage"`.

$P \to V$: Select randomness $r_{P_0} \leftarrow \{0,1\}^*$ for the original prover P_0. Send $\pi_1 = P_0(x; r_{P_0})$.

$V \to P$: Send m and decommit to all the secret shares of m, other than those decommitted in the preamble stage. Specifically, decommit to $\{m_{i,j}^{1-b_{i,j}}\}_{i,j=1}^k$.

$P \to V$: Verify that the decommitments are all valid and that $m = m_{i,j}^{1-b_{i,j}} \oplus m_{i,j}^{b_{i,j}}$, for all $i, j \in [1, k]$. If verification fails, halt and abort. Otherwise, send $\pi_2 = P_0(x, m; r_{P_0})$.

Verification stage

$V \to P$: Send the message `"start final stage"`.

$P \to V$: Send π to prove that z, sent in the initial setup stage, is a good index string.

Verifier V accepts if the original verifier V_0 accepts on (π_1, m, π_2) and $\mathsf{Verify}(x, z, \pi) = \mathtt{accept}$.

From the Protocol 5.10, we can easily derive the prover efficiency, round complexity and completeness claims of Theorem 5.8. In addition, the soundness follows from the hiding property of the concurrently-extractable commitment scheme (Lemma 3.3). This is because a cheating prover will not know the committed messages of the verifier until the verifier decommits to all secret shares of m. We summarize our results in the following lemma, deferring the proof of concurrent zero knowledge claim to Section 5.5.

Lemma 5.11. *The interactive protocol (P, V), given in Protocol 5.10, has the following properties (with the numbering consistent with Theorem 5.8):*

3. *The round complexity of (P, V) increases only by an additive factor of $\widetilde{O}(\log n)$, with n being the security parameter, compared to the original protocol (P_0, V_0).*
4. *The completeness of (P, V) is exactly the same as that of (P_0, V_0), while the soundness error increases by only a negligible additive term (as a function of n).*
5. *The prover strategy P can be implemented in probabilistic polynomial-time with oracle access to P_0. In particular, if P_0 is efficient, so is P.*

5.5 Our Simulator

We follow closely the proof structure of Lemma 4.9 in Section 4.4. The proof techniques used in this section are more complicated than the previous, hence we will maintain compatible notations whenever possible and introduce new notations when needed. We begin by summarizing the results contained in this subsection. When combined with Lemma 5.11 from the previous subsection, these results yield Theorem 5.8.

Lemma 5.12. *The interactive protocol (P, V), given in Protocol 5.10, has the following properties (with the numbering consistent with Theorem 5.8):*

1. *Zero-knowledge guarantee is preserved. That is, if (P_0, V_0) is statistical (resp., computational) zero knowledge, then (P, V) is concurrent statistical (resp., computational) zero knowledge.*
2. *Prover P is black-box simulatable in* strict *polynomial time.*

5.5.1 First simulation procedure

The prover strategy is decomposed into four parts, the index generation, preamble, main and final stages. Since the index generation, preamble and verification stages can be implemented efficiently, we will focus on the main stage prover P_{main}. Note that P_{main}'s randomness is independent of the other stages.

Concurrent adversarial sender $\widehat{S}$. We have two extra stages as compared to the previous case in Section 4.4.1, nevertheless it turns out that we will only need to slightly modify the adversarial sender $\widehat{S}$ by providing auxiliary messages[11] $[(z_s, \pi_s)]_{s\in[1,Q]}$ as additional input.

The definition of a full transcript will now include all messages from the index generation, preamble, main and verification stages. To convert preamble transcript $\mathcal{T}_{\text{pre}}$ into a full transcript, we use a similar technique as in Section 4.4.1. For every session s, we use oracle P_{main} (with prover randomness $r_{P_0} = \mathcal{F}((v_1[s], s); r_{\mathcal{F}})$) to obtain the prover main stage messages, use auxiliary inputs $[(z_s, \pi_s)]_{s\in[1,Q]}$ to obtain prover messages in the index generation and verification stages (namely z_s and π_s for each session s), and use V^* to determine the verifier's main stage messages as well as the scheduling of the messages. A formal definition of $\widehat{S}$ is given next.

[11] These auxiliary messages are used for the index generation and verification stages of the witness-binding commitments (refer to Definition 5.2).

$\widehat{S}^{P_{\text{main}},V^*}(\mathcal{T}_{\text{pre}}; r_{\mathcal{F}}, [(z_s,\pi_s)]_{s\in[1,Q]})$:

1. Let $\mathcal{T}_{\text{pre}} = ((s_1,v_1),p_1,(s_2,v_2),p_2,\ldots,(s_t,v_t),p_t)$.
2. Initialize $\mathcal{T}_{\text{full}} = ()$ and $j = 1$.
3. Query oracle V^* on $\mathcal{T}_{\text{full}}$ to obtain $(s,v) = V^*(\mathcal{T}_{\text{full}})$. Depending on the value of v, s, and t, do the following.

   Case 1: v is a preamble stage message and $j = t+1$. In this case, send (s,v).

   Case 2: $(s,v) = (\texttt{end},\alpha)$ and $j = t+1$. In this case, output α and halt.

   Case 3: v is a preamble stage message, $j \le t$, and $(s,v) = (s_j,v_j)$. In this case, update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s_j,v_j),p_j)$ and $j = j+1$. Repeat Step 3.

   Case 4: v is a main stage message. Proceed as follows.

   (a) Let $T[s]$ denote all the messages of session s in $\mathcal{T}_{\text{full}}$, and let $v_1[s]$ be the verifier's first preamble message in $T[s]$.
   (b) Set the main stage prover's randomness $r_{P_0} = \mathcal{F}((v_1[s],s); r_{\mathcal{F}})$.
   (c) Query oracle P_{main} to obtain $p = P_{\text{main}}(x, T[s]\circ v; r_{P_0})$.
   (d) Update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s,v),p)$, and repeat Step 3.

   Case 5: $v =$ `"start session"`. In this case, update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s,v),z_s)$, and repeat Step 3.

   Case 6: $v =$ `"start final stage"`. In this case, update $\mathcal{T}_{\text{full}} = \mathcal{T}_{\text{full}} \circ ((s,v),\pi_s)$, and repeat Step 3.

   Case 7: Otherwise, halt and output `fail`, indicating that $\mathcal{T}_{\text{pre}}$ was not a valid preamble transcript.

The main difference between $\widehat{S}$, as defined above, compared to that defined previously in Section 4.4.1 is the additional handling of the index generation and verification stages (refer to Cases 5 and 6 above).

First simulator. Similar to what we did in Section 4.4.1, we write $\widehat{S}[r_{\mathcal{F}}, [(z_s,\pi_s)]](\cdot)$ to represent the deterministic oracle $\widehat{S}^{P_{\text{main}},V^*}(\cdot; r_{\mathcal{F}}, [(z_s,\pi_s)]_{s\in[1,Q]})$. Our first (inefficient) simulator is very similar in structure.

$\textsf{Sim-One}^{P_{\text{main}},V^*}(x; r_{\mathcal{F}}, [(z_s,\pi_s)]_{s\in[1,Q]})$:

1. Set $\mathcal{COM} = \{\textsf{Com}_1, \textsf{Com}_2, \ldots, \textsf{Com}_Q\}$, where $\textsf{Com}_s = \textsf{WB-Com}_{x,z_s}$ (the witness-binding problem-dependent commitment) for all $s \in \{1,2,\ldots,Q\}$.
2. Output $\textsf{CEC-Sim}^{\widehat{S}[r_{\mathcal{F}},[(z_s,\pi_s)]]}(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$.

As done in Section 4.4, we simplify notations by writing $\textsf{CEC-Sim}^{\widehat{S}[r_{\mathcal{F}},[(z_s,\pi_s)]]}$ to represent $\textsf{CEC-Sim}^{\widehat{S}[r_{\mathcal{F}},[(z_s,\pi_s)]]}(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$, and avoid repeating common inputs $(\mathcal{COM}, 1^\ell, 1^k, 1^n, 1^Q)$. We write $[(z_s,\pi_s)] \leftarrow \textsf{Generate}(x)$ to denote taking Q independent samples, one for each pair (z_s,π_s), from the probabilistic algorithm $\textsf{Generate}(x)$. We have the following lemma.

Lemma 5.13. *If the number of concurrent sessions Q is bounded by $\text{poly}(|x|)$, then output of the first simulator $\left\{\textsf{Sim-One}^{P_{\text{main}},V^*}(x)\right\}_{x\in\Pi_{\mathrm{Y}}}$ (over random coin tosses $r_{\mathcal{F}} \leftarrow \{0,1\}^*$ and $[(z_s,\pi_s)] \leftarrow \textsf{Generate}(x)$) is statistically indistinguishable from $\left\{\langle P, V^*\rangle(x)\right\}_{x\in\Pi_{\mathrm{Y}}}$.*

The proof of the above lemma is similar to the proof of Lemma 4.10 and hence omitted.

5.5.2 Second simulation procedure.

We will define a *stateful* probabilistic oracle $\mathcal{O}_{P(w)}[\mathsf{state}]$, with w being any valid witness of $x \in \Pi_Y$, to substitute oracle P_{main} and the exponentially long random key $r_{\mathcal{F}}$. Unlike Section 4 where we had perfectly-binding problem dependent commitments, stateful oracles are needed because witness-binding commitments are not necessarily binding (even when $x \in \Pi_Y$). The state of the oracle, represented as an array, is used to record the coin tosses of the CVZK simulator S_{CVZK}, which will be needed by the witness-completable simulator S_{wc} to complete the simulation (in the case when the commitments are not binding).

Stateful probabilistic oracle $\mathcal{O}_{P(w)}[\mathsf{state}](q)$: There are two types of queries.

Query $q = m$ (where m is a possible verifier message):
Let j denote the least integer such that $\mathsf{state}[j] = \perp$. Select randomness $r \leftarrow \{0,1\}^*$, and set $\pi_1 = P_0(x, w, r)$ and $\pi_2 = P_0(x, w, m, r)$. Output (π_1, m, π_2, j). Update the state by setting $\mathsf{state}[j] = (r, m)$.

Query $q = (m', i)$:
If $\mathsf{state}[i] = \perp$, output $\perp$. Else, let $(r, m) = \mathsf{state}[i]$ and set $\pi_2' = P_0(x, w, m', r)$. Output π_2'.

The initial state of the oracle is the empty array, *i.e.,* $\mathsf{state}[j] = \perp$ for all j.

Using the same argument as in Section 4.4.1, we now observe that with the recorded main stage prover responses $\mathsf{Prover\text{-}Msg}$, we can mimic the execution of $\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}},[(z_s,\pi_s)]]}$ (in Section 5.5.1) without using oracle P_{main} or the exponentially long random key $r_{\mathcal{F}}$. Specifically, instead of querying P_{main} to obtain $p = P_{\text{main}}(x, T[s] \circ v; \mathcal{F}((v_1[s], s); r_{\mathcal{F}}))$, we let $(\pi_1, m, \pi_2, j) = \mathsf{Prover\text{-}Msg}[(v_1[s], s)]$ and use π_1 and π_2 as the first and second main stage prover responses, respectively.

However, because the commitments are no longer perfectly binding, we will need to employ the use oracle $\mathcal{O}_{P(w)}$ to supply the second main stage prover message π_2 when the verifier's message differs from the extracted message m provided by $\mathsf{CEC\text{-}Sim}$. In this case, we query $\mathcal{O}_{P(w)}(m', j)$ to obtain the prover's response to m'.

$\mathsf{Sim\text{-}Two}^{\mathcal{O}_{P(w)},V^*}(x, [(z_s, \pi_s)]_{s\in[1,Q]})$**:**

1. *(Similar to* $\mathsf{Sim\text{-}One}$*)* Set $\mathcal{COM} = \{\mathsf{Com}_1, \mathsf{Com}_2, \ldots, \mathsf{Com}_Q\}$, where $\mathsf{Com}_s = \mathsf{WB\text{-}Com}_{x,z_s}$ (the witness-binding problem-dependent commitment) for all $s \in [1, Q]$.
2. Initialize the array of main stage prover responses $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] = \perp$ for all pairs $(v_1[s], s)$.
3. Without using oracle P_{main} and randomness $r_{\mathcal{F}}$, run $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$ until $\mathsf{CEC\text{-}Sim}$ queries $\widehat{S}$ on some $\mathcal{T}_{\text{pre}}$. When this happens, do the following:
   (a) Freeze the execution of $\mathsf{CEC\text{-}Sim}^{\widehat{S}}$.
   (b) Let $T[s]$ denote all the preamble stage messages of session s in $\mathcal{T}_{\text{pre}}$, and let $v_1[s]$ be the verifier's first (preamble) message in $T[s]$. If $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] \neq \perp$, then let $(\pi_1, m, \pi_2, j) = \mathsf{Prover\text{-}Msg}[(v_1[s], s)]$, and proceed to Step 3d.
   (c) For every s such that $T[s]$ is a valid commit phase transcript, $\mathsf{CEC\text{-}Sim}$ provides an extracted message m. Using m, query $\mathcal{O}_{P(w)}(m)$ to obtain the prover responses (π_1, m, π_2, j). Update $\mathsf{Prover\text{-}Msg}[(v_1[s], s)] = (\pi_1, m, \pi_2, j)$.

(d) Use (π_1, m, π_2, j) and possibly an additional query to $\mathcal{O}_{P(w)}$ to answer queries to oracle P_{main} instead of querying P_{main} directly. Specifically, do the following:
- If the query is for the first prover's main stage message, answer the query with π_1.
- If the query is for the second prover's main stage message in response to verifier's message m, answer the query with π_2.
- If the query is for the second prover's main stage message in response to a different verifier's message $m' \neq m$, query $\mathcal{O}_{P(w)}(m', j)$ to obtain π_2' and answer the query with π_2'.

(e) Continue the execution of $\mathsf{CEC\text{-}Sim}^{\widehat{S}[r_{\mathcal{F}}, [(z_s, \pi_s)]]}$ as in Step 3.

As in the case of the second simulation procedure in Section 4.4.2, $\mathsf{Sim\text{-}Two}$ acts exactly like $\mathsf{Sim\text{-}One}$.

Lemma 5.14. *For all $x \in \Pi_{\mathrm{Y}}$, all valid witness w of x, and all $[(z_s, \pi_s)]_{s \in [1,Q]}$, we have that*

$$\left[\mathsf{Sim\text{-}One}^{P_{\text{main}}, V^*}(x, r_{\mathcal{F}}, [(z_s, \pi_s)])\right]_{r_{\mathcal{F}} \leftarrow \{0,1\}^*} \equiv \mathsf{Sim\text{-}Two}^{\mathcal{O}_{P(w)}, V^*}(x, [(z_s, \pi_s)]).$$

Furthermore, $\mathsf{Sim\text{-}Two}$ *runs in polynomial time when given access to oracles $\mathcal{O}_{P(w)}$ and V^*.*

A notable difference in the execution of $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{P(w)}, V^*}(x, [(z_s, \pi_s)]_{s \in [1,Q]})$ compared to the previous $\mathsf{Sim\text{-}Two}$ in Section 4.4.2 is the use of witness-binding commitments (instead of perfectly-binding ones), hence necessitating an augmentation of $\mathcal{O}_P$ to the stateful oracle $\mathcal{O}_{P(w)}$. Otherwise, the proof of the above lemma is similar to the proof of Lemma 4.11 in Section 4.4.2, and hence omitted.

5.5.3 Replacing the oracle $\mathcal{O}_{P(w)}$

We are given that (P_0, V_0) is a wCVZK protocol. Let S_{CVZK} be the CVZK simulator and S_{wc} be the witness-completable simulator for (P_0, V_0). Let w be any valid witness of instance $x \in \Pi_{\mathrm{Y}}$. We note that unlike Section 4.4.3, we still use a witness since S_{wc} requires it. In order to replace oracle $\mathcal{O}_{P(w)}[\mathsf{state}]$ with these simulators, we define a new stateful probabilistic oracle (that uses these simulators as subroutine).

Stateful probabilistic oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}[\mathsf{state}](q)$: There are two types of queries.

Query $q = m$:
Let j denote the least integer such that $\mathsf{state}[j] = \perp$. Select randomness $r \leftarrow \{0,1\}^*$, and set $(\pi_1, m, \pi_2) = S_{\mathrm{CVZK}}(x, m, r)$. Output (π_1, m, π_2, j). Update the state by setting $\mathsf{state}[j] = (r, m)$.

Query $q = (m', i)$:
If $\mathsf{state}[i] = \perp$, output $\perp$. Else, let $(r, m) = \mathsf{state}[i]$ and set $\pi_2' = S_{\mathrm{wc}}(x, m', w, m, r)$. Output π_2'.

The initial state of the oracle is the empty array, *i.e.,* $\mathsf{state}[j] = \perp$ for all j.

The next lemma states that the oracles $\mathcal{O}_{\mathrm{wCVZK}(w)}$ and $\mathcal{O}_{P(w)}$ are indistinguishable to circuits that queries its oracle at most

Lemma 5.15. *If (P_0, V_0) is a computational (resp., statistical) wCVZK protocol, then for all polynomial-size*[12] *circuit D, the ensembles $\left\{D^{\mathcal{O}_{\mathrm{wCVZK}(w)}}(x)\right\}_{(x,w)\in \mathrm{R}_\Pi}$ and $\left\{D^{\mathcal{O}_{P(w)}}(x)\right\}_{(x,w)\in \mathrm{R}_\Pi}$ are computationally (resp., statistically) indistinguishable.*

Proof. Assume that circuit $D^{\mathcal{O}}(x)$ queries its oracle at most $t \leq \mathrm{poly}(|x|)$ times. For each oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}$ and $\mathcal{O}_{P(w)}$, we define the following auxiliary oracles $(\mathcal{A}_j)_{1\leq j\leq t}$ and $(\mathcal{B}_j)_{1\leq j\leq t}$, respectively.

Oracle $\mathcal{A}_j(m)$: Begin with oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}[\mathsf{state}]$ in the following state: $\mathsf{state}[j] = \perp$, and for all $i < j$, $\mathsf{state}[i] \neq \perp$.

First query: Output $\mathcal{O}_{\mathrm{wCVZK}(w)}(m)$.

Subsequent queries: Output $\mathcal{O}_{\mathrm{wCVZK}(w)}((m, j))$.

Oracles $(\mathcal{B}_j)_{1\leq j\leq t}$ are defined similarly, with every occurrence of $\mathcal{O}_{\mathrm{wCVZK}(w)}$ substituted with $\mathcal{O}_{P(w)}$.

Oracle $\mathcal{B}_j(m)$: Begin with oracle $\mathcal{O}_{P(w)}[\mathsf{state}]$ in the following state: $\mathsf{state}[j] = \perp$, and for all $i < j$, $\mathsf{state}[i] \neq \perp$.

First query: Output$\mathcal{O}_{P(w)}(m)$.

Subsequent queries: Output $\mathcal{O}_{P(w)}((m, j))$.

Observe that by the definition of statistical wCVZK (Definition 5.5), for every $1 \leq j \leq t$, we have that for all circuit D,

$$D^{\mathcal{A}_j}(x) \approx_{\mathrm{s}} D^{\mathcal{B}_j}(x). \tag{1}$$

This is because the first query to $\mathcal{A}_j$ corresponds to $(S_{\mathrm{CVZK}}(x, m, \tilde{r}), j)$, and the subsequent queries correspond to $S_{\mathrm{wc}}(x, w, m, \cdot, \tilde{r})$. Likewise, the first query to $\mathcal{B}_j$ corresponds to $(\mathrm{view}_{V_m}^{P(w,r)}(x), j)$, and the subsequent queries correspond to $P_2(x, w, \cdot, r)$.

By virtue of the way we defined the auxiliary oracles, the t oracles $(\mathcal{A}_1, \ldots, \mathcal{A}_t)$ can perfectly mimic the oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}$. Specifically, asking $\mathcal{O}_{\mathrm{wCVZK}(w)}$ a query of the type $q = m$ for the j-th time is equivalent to querying oracle $\mathcal{A}_j$ on m. In addition, asking a query of the type $q = (m', i)$ is equivalent to querying oracle $\mathcal{A}_i$ on m' (if we have previously queried $\mathcal{A}_i$), or to answering $\perp$ (if we have not previously queried $\mathcal{A}_i$). The same argument applies to oracle $\mathcal{O}_{P(w)}$ and its t auxiliary oracles $(\mathcal{B}_1, \ldots, \mathcal{B}_t)$.

Therefore, it remains to be shown that for all circuit D,

$$D^{\mathcal{A}_1\mathcal{A}_2\cdots\mathcal{A}_t}(x) \approx_{\mathrm{s}} D^{\mathcal{B}_1\mathcal{B}_2\cdots\mathcal{B}_t}(x).$$

The above statement can be proven using a standard hybrid argument. Specifically, consider a new distinguisher $\tilde{D}^{\mathcal{O}} \stackrel{\mathrm{def}}{=} D^{\mathcal{A}_1\cdots\mathcal{A}_{j-1}\mathcal{O}\mathcal{B}_{j+1}\cdots\mathcal{B}_t}$, noting that $\tilde{D}$ is only a polynomial factor bigger than D since the auxiliary oracles, $\mathcal{A}_j$'s and $\mathcal{B}_j$'s, are be implementable by a polynomial-sized circuit (with witness w hardwired).

[12] In the case of statistically indistinguishability, we need not restrict D to be polynomial-size, but it has to query its oracle at most polynomial times. For the purposes of this paper, D is the simulator $\mathsf{Sim\text{-}Two}^{\mathcal{O},V^*}$, which runs in polynomial time (assuming V^* does too).

For all $1 \leq j \leq t$, we have that

$$\begin{aligned} D^{\mathcal{A}_1 \cdots \mathcal{A}_{j-1} \mathcal{A}_j \mathcal{B}_{j+1} \cdots \mathcal{B}_t}(x) &= \tilde{D}^{\mathcal{A}_j}(x) \\ &\approx_{\mathrm{s}} \tilde{D}^{\mathcal{B}_j}(x) \quad \text{[from Equation (1)]} \\ &= D^{\mathcal{A}_1 \cdots \mathcal{A}_{j-1} \mathcal{B}_j \mathcal{B}_{j+1} \cdots \mathcal{B}_t}(x). \end{aligned}$$

This completes our proof of Lemma 5.15. □

Observe that Sim-Two runs in polynomial-time (with oracle queries), hence it can only query its oracles polynomial times. By Lemma 5.15, we can replace oracle $\mathcal{O}_{P(w)}$ with $\mathcal{O}_{\mathrm{wCVZK}(w)}$ and obtain an indistinguishable simulation. That is for all fixed $[(z_s, \pi_s)]_{s \in [1,Q]}$ and valid witness w,

$$\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)}, V^*}(x, [(z_s, \pi_s)]) \approx \mathsf{Sim\text{-}Two}^{\mathcal{O}_{P(w)}, V^*}(x, [(z_s, \pi_s)])$$

Consequently, the above fact together with Lemmas 5.13 and 5.14 gives us the following.

Lemma 5.16. *If (P_0, V_0) is a computational (resp., statistical) wCVZK protocol, then the ensembles $\left\{ \mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)}, V^*}(x) \right\}_{(x,w) \in \mathrm{R}_\Pi}$ and $\left\{ \langle P, V^* \rangle (x) \right\}_{x \in \Pi_{\mathrm{Y}}}$ are computationally (resp., statistically) indistinguishable.*

5.5.4 Obtaining a valid witness

The simulator $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)}, V^*}(x, [(z_s, \pi_s)])$ is still not efficiently implementable because it requires a valid witness w for the oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}$. Note however that $\mathcal{O}_{\mathrm{wCVZK}(w)}$ needs a valid witness only when it needs to answer a query of the form (m', i). This happens when V^* breaks the binding property of the witness-binding commitments, in which case, we should be able to extract a witness (see Definition 5.2). Specifically, we propose the following witness extraction procedure.

$\mathsf{Witness\text{-}Extractor}(x)$:

1. Select a random index $j \leftarrow [1, Q]$. Select $r_E \leftarrow \{0,1\}^*$, and set $z_j = \mathsf{Simulate}(x; r_E)$ and $\pi_j = \varepsilon$ (where ε represents an empty string). For $k \neq j$, set $(z_k, \pi_k) \leftarrow \mathsf{Generate}(x)$.
2. With the selected values of $[(z_s, \pi_s)]$, run $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)}, V^*}(x, [(z_s, \pi_s)]_{s \in [1,Q]})$, with an empty witness $w = \varepsilon$ (since we do not need a valid witness to begin the simulation).
3. If during the execution, Sim-Two asks oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}$ a query of the form (m', i), it is the case that we can find two decommitments in some session s that is associated with the same commitment.[13] Specifically, let the decommitments $d_0 = (x, z_s, 0; r_C)$ and $d_1 = (x, z_s, l; r'_C)$ be such that $\mathsf{WB\text{-}Com}_{x,z_s}(0; r_C) = \mathsf{WB\text{-}Com}_{x,z_s}(1; r'_C)$. If $s = j$, then output $w = \mathsf{Extract}(x, r_C, r'_C, r_E)$. Otherwise, halt and abort.

It is clear that $\mathsf{Witness\text{-}Extractor}(x)$ runs in polynomial time since Sim-Two and all the witness-binding commitment algorithms $(\mathsf{Generate}, \mathsf{WB\text{-}Com}, \mathsf{Verify}, \mathsf{Simulate}, \mathsf{Extract})$ run in polynomial time. In addition, $\mathsf{Witness\text{-}Extractor}(x)$ also produces a witness with a reasonable probability, as shown in the next lemma.

Lemma 5.17. *Let p denote the probability that $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)}, V^*}(x, [(z_s, \pi_s)])$ (over $[(z_s, \pi_s)] \leftarrow \mathsf{Generate}(x)$) asks oracle $\mathcal{O}_{\mathrm{wCVZK}(w)}$ a query of the form (m', i). The algorithm $\mathsf{Witness\text{-}Extractor}(x)$ outputs a valid witness of x with probability greater or equal to p/Q.*

[13] This is due to the binding property of the concurrently-extractable commitments (Lemma 3.4), and the fact that simulator CEC-Sim gives a compatible message M_s for each valid commit phase transcript $T[s]$.

Proof. We compare the following two executions:

(A) The execution of $\mathsf{Witness\text{-}Extractor}(x)$.

(B) The execution of $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)},V^*}(x,[(z_s,\pi_s)])$ when it is given a valid witness w and $[(z_s,\pi_s)] \leftarrow \mathsf{Generate}(x)$.

Since taking $z_j \leftarrow \mathsf{Simulate}(x)$ has the same exact distribution as the first component of $\mathsf{Generate}(x)$ (Property 3 in Definition 5.2) and the witness w does not matter until a query of the form (m', i) is asked, (A) and (B) will execute in a similar manner until the first query of that form (m', i) is asked. If and when this happens, there is a probability of $1/Q$ that our guess j will equal the current session s. (The probability of $1/Q$ is due to the fact that j is independent of the event that a query of the form (m', i) is asked, and of the first session s in which such a query is asked.)

If we guessed the right session $j = s$, then $w = \mathsf{Extract}(x, r_C, r'_C, r_E)$ must be a valid witness of x (Property 4 in Definition 5.2). Therefore, the probability of obtaining a valid witness is greater or equal to $(1/Q) \cdot p = p/Q$. $\square$

5.5.5 Final simulator

Consider the following mental experiment: Run $\mathsf{Witness\text{-}Extractor}(x)$ repeatedly (with each run done with independent randomness) until we obtain a valid witness w. With w in hand, select $[(z_s,\pi_s)] \leftarrow \mathsf{Generate}(x)$ and output $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)},V^*}(x,[(z_s,\pi_s)])$.

Because witness w is generated independently from the rest of the execution, the output of the mental experiment will be statistically indistinguishable from $\langle P, V^*\rangle(x)$ (by Lemma 5.16). However, the mental experiment's running time could be exponential (because we expect to run $\mathsf{Witness\text{-}Extractor}(x)$ Q/p times, where p is the probability that the verifier breaks the commitment in the execution of $\mathsf{Sim\text{-}Two}$).

To overcome this problem, our final simulator will only use the witness extraction algorithm when needed.

$\mathsf{Sim\text{-}Three}^{V^*}(x)$**:**

1. Select $[(z_s,\pi_s)] \leftarrow \mathsf{Generate}(x)$ and output $\mathsf{Sim\text{-}Two}^{\mathcal{O}_{\mathrm{wCVZK}(w)},V^*}(x,[(z_s,\pi_s)])$.
2. If a valid witness is needed (to answer queries of the form (m', i)), do the following:
   (a) Freeze the execution of $\mathsf{Sim\text{-}Two}$.
   (b) Run $\mathsf{Witness\text{-}Extractor}(x)$ repeatedly (with each run done with independent randomness) until we obtain a valid witness w.
   (c) Using witness w, continue the frozen execution of $\mathsf{Sim\text{-}Two}$ and output it.

Clearly, $\mathsf{Sim\text{-}Three}^{V^*}(x)$'s output is identical to that of the mental experiment, hence it is statistically indistinguishable from $\langle P, V^*\rangle(x)$. Its expected running time is stated in the following lemma.

Lemma 5.18. *The simulator* $\mathsf{Sim\text{-}Three}^{V^*}$ *runs in expected polynomial-time.*

Proof. Let p be the probability that the execution of Step 1 in the description of $\mathsf{Sim\text{-}Three}^{V^*}(x)$ will require a valid witness (to answer queries of the form (m', i)). When Step 1 does not require a witness, the running time is $\mathrm{poly}(|x|)$.

In the case when Step 1 requires a witness, we will need to run Step 2. By Lemma 5.17, $\mathsf{Witness\text{-}Extractor}(x)$ outputs a valid witness with probability greater or equal to p/Q. We analyze the expected total running time as follows.

$$\begin{aligned}
\mathbf{E}[\text{running time of } \mathsf{Sim\text{-}Three}^{V^*}(x)] &= \Pr[\text{Step 1 requires a witness}] \cdot \mathbf{E}[\text{running time of Step 2}] \\
&\quad + \mathrm{poly}(|x|) \\
&\leq p \cdot ((Q/p) \cdot \mathrm{poly}(|x|)) + \mathrm{poly}(|x|) \\
&\leq \mathrm{poly}(|x|),
\end{aligned}$$

with the final inequality following from the fact that the number of concurrent sessions $Q \leq \mathrm{poly}(|x|)$. □

We have shown that Sim-Three is an expected polynomial-time black-box simulator for Protocol 5.10. This proves Lemma 5.12, thus completing our proof of Theorem 5.8.

6 Acknowledgements

We thank Alon Rosen for helpful discussions.

References

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP $\cap$ coNP. In *Proc. 45th FOCS*, pages 362–371, 2004.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115, 2001.

[BCC88] Gilles Brassard, David Chaum, and Crepeau Crepeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BMO90] M. Bellare, S. Micali, and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proc. 22nd STOC*, pages 482–493, 1990.

[CKPR03] Ran Canetti, Joe Kilian, Erez Petrank, and Rosen Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM Journal on Computing*, 32(1):1–47, 2003.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proc. CRYPTO '98*, pages 13–25, 1998.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[DDN01] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2001.

[DiC00] Giovanni Di Crescenzo. Removing complexity assumptions from concurrent zero-knowledge proofs. In *COCOON: Annual International Conference on Computing and Combinatorics*, pages 426–435, 2000.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proc. 30th STOC*, pages 409–418, 1998.

[ES02] Edith Elkind and Amit Sahai. A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. Cryptology ePrint Archive, Report 2002/042, 2002. `http://eprint.iacr.org/`.

[Fei90] Uriel Feige. *Alternative models for zero knowledge interactive proofs*. PhD thesis, Weizmann Institute of Science, Israel, 1990.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

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

[GMR98] Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *Proc. of the 5th ACM Conference on Computer and Communications Security (CCS-98)*, pages 67–72, 1998.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[Gol01] Oded Goldreich. *Foundations of cryptography*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[Gol02] Oded Goldreich. Zero-knowledge twenty years after its invention. `http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html`, 2002.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proc. 30th STOC*, pages 399–408, 1998.

[GV99] Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *Proc. 14th IEEE Conference on Computational Complexity*, pages 54–73, 1999.

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IOS97] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *Journal of Cryptology*, 10(1):37–49, 1997.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *Proc. 33rd STOC*, pages 560–569, 2001.

[KPR98] J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the Internet. In *Proc. 39th FOCS*, pages 484–492, 1998.

[MP03] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *Proc. EUROCRYPT '03*, pages 140–159, 2003.

[MV03] Daniele Micciancio and Salil Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In *Proc. CRYPTO '03*, pages 282–298, 2003.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. In *Proc. 22nd STOC*, pages 427–437, 1990.

[Oka00] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *Journal of Computer and System Sciences*, 60(1):47–108, 2000.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, 1991.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Second Israel Symposium on Theory of Computing Systems*, pages 3–17, 1993.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *Proc. 43rd FOCS*, pages 366–375, 2002.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *Proc. EUROCRYPT '99*, pages 415–431, 1999.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *Proc. CRYPTO '00*, pages 451–468, 2000.

[Ros03] Alon Rosen. *The Round-Complexity of Black-Box Concurrent Zero-Knowledge*. PhD thesis, Weizmann Institute of Science, Israel, 2003.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th FOCS*, pages 543–553, 1999.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2), 2003.

[Vad04] Salil Vadhan. An unconditional study of computational zero knowledge. In *Proc. 45th STOC*, pages 176–185, 2004.

[Yao86] Andrew C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.