Precise Zero Knowledge

Silvio Micali* Rafael Pass†

December 1, 2007‡

Abstract

We put forward the notion of *Precise Zero Knowledge* and provide its first implementations in a variety of settings under standard complexity assumptions. Whereas the classical notion of Zero Knowledge bounds the knowledge of a player in terms of his *potential* computational power (technically defined as polynomial-time computation), Precise Zero Knowledge bounds the knowledge gained by a player in terms of its *actual* computation (which can be considerably less than any arbitrary polynomial-time computation).

*CSAIL, MIT, E-Mail: silvio@csail.mit.edu

†Department of Computer Science, Cornell University, E-Mail: rafael@cs.cornell.edu

‡A preliminary version of this paper appeared in *38'th STOC*, 2006 under the name *Local Zero Knowledge*. This version contains a slight revision from August 2011.

Contents

1 Introduction

The works of Goldwasser and Micali [36] and of Goldwasser, Micali and Rackoff [37] put forward a computational approach to knowledge in interactive systems. In a nutshell, their approach can be summarized as follows:

> *A player knows only what he can feasibly compute.*

Since "feasible computation" is formalized as probabilistic polynomial-time computation, their notion bounds the knowledge gained by a player in an interaction in terms of what is computable in probabilistic polynomial time.

In this paper, we put forward a stronger notion that *precisely* bounds the knowledge gained by a player in terms of the *actual* computation he has performed (which can be considerably less than any arbitrary polynomial-time computation). We focus our treatment on interactive proofs [37], but as we shall see, our notion naturally extends to more general types of interactions—in particular, secure encryption schemes and general secure multi-party computations.

1.1 Precise Zero-Knowledge

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali and Rackoff [37] are fascinating (and seemingly paradoxical) constructs, allowing one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the Verifier.

Goldwasser, Micali and Rackoff's definition essentially states that an interactive proof of $x \in L$ provides *zero* (additional) *knowledge* to the Verifier, if, for any probabilistic polynomial-time verifier V, the view of V in the interaction can be "indistinguishably reconstructed" by a probabilistic polynomial-time simulator S—interacting with no one—on just input x. The rational behind this definition is that since whatever V "sees" in the interaction can be reconstructed in polynomial-time, the interaction does not yield anything to V that cannot already be computed in polynomial-time. Thus, zero-knowledge proofs, although conveying a lot of *information* to the Verifier, guarantee that

> *The class of probabilistic polynomial-time verifiers learn nothing new from the interaction.*

We wish to consider a notion of zero knowledge that guarantees that also individual verifiers, rather than the class of polynomial-time verifiers, learn nothing new. A major step towards this goal was taken already by Goldreich, Micali and Wigderson [34], and Goldreich [31]. The refinement of [34, 31], called *knowledge tightness*, calls for a tighter coupling between the running-time of V and the running-time of S. Roughly speaking, a proof is zero knowledge with tightness $p(\cdot)$ if the expected running time of $S(x)$ is upper-bounded by $p(|x|)$ times the running time of $V(x)$ (plus some small additive polynomial in $|x|$). Think for instance of $t(n) = 2$; knowledge-tight zero knowledge then roughly says that, given any instance x, if $V(x)$'s running-time on input x is t, $S(x)$'s expected running needs to be bounded by (roughly) $2t$.

Recall that as $V(x)$ is an interactive machine, its running-time may depend on the messages it receives. In the definition of knowledge tightness, the running-time of $V(x)$ is taken to mean the *worst-case* (i.e., maximum) running-time of $V(x)$ in *any* interaction. The tightness of a zero-knowledge proof thus bounds the knowledge gained by the Verifier in terms of its *maximum* running-time in any interaction.

Our notion of Precise Zero Knowledge aims at bounding the knowledge of a verifier in an *execution-by-execution* manner. To motivate it, consider a malicious verifier V that, on input an instance $x \in \{0,1\}^n$, with probability .01 *over the messages it receives*, takes n^{50} computational steps and n steps the rest of the time. The worst-case running time[1] of V is n^{50}, and thus zero knowledge with optimal tightness only requires that V be simulated in expected time $O(n^{50})$. Does this really mean that it is indifferent for V to get out and interact with the Prover or to stay home and run S? Unless we are confident that n^{50} steps convey absolutely no knowledge (or unless we have stronger guarantees about the behavior of the simulator) the answer is no. In fact, by interacting with the Prover, V will almost always execute n steps of computation, while (in absence of extra guarantees) running the simulator might *always* cause him to invest n^{50} steps of computation! (This is not just a theoretical worry or an artifact of the definition: it actually occurs for classical protocols and simulators.[2])

Rather than preserving just the worst-case running-time of the verifier, we wish the simulation to preserve the running-time *distribution* of the verifier. In fact, we want even more: informally,

> *P provides a zero-knowledge proof of $x \in L$ if the view v of any verifier V in an interaction with P about x can be reconstructed—on just input x—in the same time (within, say, a constant factor) as that taken by V* ***in the particular view*** *v.*

In other words, whatever V can "see with P" in t steps of computation, he can reconstruct by himself in—say—$2t$ steps. We call a proof satisfying the above property a *precise zero-knowledge proof* (or a zero-knowledge proof with linear *precision*); more generally, the proof is said to have precision $p(n,t)$ if the time it takes to reconstruct a view in which V takes t steps is bounded by $p(|x|,t)$. Roughly speaking, we formalize this notion as follows: for any Verifier V, we require the existence of a simulator S such that for any true statement x and any view v output by $S(x)$ (on input some random tape r), if V takes t steps in the view[3] v, then $S(x)$ cannot have taken more than $p(|x|,t)$ steps to generate it (using the random tape r).

In essence, whereas knowledge-tight zero-knowledge only requires that the simulation incurs a small slow down relative to the the maximum running time of the verifier, our notion calls for a small slow down with respect to the actual running time in each *particular* execution.

Notice that if restricting our attention to verifiers that *always* execute the same (or roughly the same) number of computation steps, then any standard zero-knowledge simulation with tightness $O(1)$ also has precision $p(n,t) = O(t) + \text{poly}(n)$. However, if considering *general* verifiers whose running-time depend (in some non-trivial fashion) on the instance and the messages received, then precise simulation seems harder to obtain. (Indeed, as we shall see, known simulations do not even guarantee precision $p(n,t) = \text{poly}(n,t)$.)

Let us provide some applications of the new notion.

[1] In fact, also the expected running-time of V is $O(n^{50})$.

[2] Consider for instance, the protocol of Feige-Shamir [28] when instantiated with Goldreich, Micali and Wigderson's Graph 3-Coloring proof system [34]. As above, consider a verifier V that with prob .01 runs in time n^{50}, and otherwise in time n. The Feige-Shamir simulator runs the verifier n^2 times, each time feeding it new messages (this is done in order to extract a "fake" witness). The probability that the verifier runs in time n in all n^2 rewindings is $.99^{n^2}$. Thus, although the verifier only runs in time n^{50} with probability .01, the simulator will essentially always run in time $O(n^{50})$.

[3] Recall that the view of V contains both the messages received by V and its random tape; given these, the execution of V is deterministic and we can thus determine the number of steps it takes in this view.

Preserving Success/Time-Distribution. Standard Zero-Knowledge proofs provide the guarantee that the Verifier will not be able to compute any properties of the statement proved, that cannot already be computed (without interacting with the Prover) in expected time that is comparable to the worst-case running-time of the Verifier. Barak and Lindell [7] point out that such a coupling between the *expected* running-time of the simulator and the *worst-case* running-time of the Verifier, allows for a success-probability/running-time trade-off for the Verifier: a malicious Verifier might potentially with probability, say, $\frac{1}{100}$ compute some property after 1 year, that would have taken him 100 years to compute before the interaction. They also note that zero-knowledge proofs with *strict polynomial-time* simulators, i.e., simulators whose worst-case running-time is coupled to the worst-case running-time of the Verifier do not allow such a trade-off. In other words, strict polynomial-time zero-knowledge proofs do not allow for success-probability/(worst-case) running-time trade-offs.

Precise Zero Knowledge additionally guarantees that the success-probability/*running-time distribution* of the Verifier is preserved. More precisely, consider a verifier that can compute some property with a certain success probability and a running time that is specified by some probability distribution (over the random coins of both the honest prover and the verifier). Then, the notion of Precise Zero Knowledge guarantees that the same property can be computed with (roughly) the same success probability and running-time distribution, without the help of the prover.

"More Deniable" Identification. Perhaps the most important use of zero-knowledge proofs in practice consists of (1) convincing a gate keeper of our identity, without (2) leaving any evidence of our interaction that can be believed by a third party [29, 18, 23]. Intuitively, the information left in the hands of the gate keeper consists of his view of the interaction, something that he could reconstruct himself without any help in the case of a zero knowledge proof (of $x \in L$, for a fixed hard language L and for an input x publicly linked to the identity). However, let us argue that also for such a "deniable identification" application, standard definitions of zero-knowledge might not directly stipulate sufficient security guarantees.

Consider a gate keeper that with probability 0.01 takes n^{50} steps and n steps otherwise. With probability 0.99, such a gate keeper might obtain in n steps a view of the interaction that would have taken him n^{50} steps to generate, *if naively running the zero-knowledge simulator*. Such a view might therefore serve as a (plausible) evidence of the authenticity of his interaction with us. Of course, if we know that the gate keeper always executes either n steps, or n^{50} steps, then any view obtained by the keeper in n steps can be reconstructed in roughly n steps by considering the zero-knowledge simulator for a "truncated" version of the keeper, that executes at most n steps. In general, however, the running time of the gate keeper might be much more erratic (for instance, the gate keeper might take $\frac{1}{p}$ steps with probability p, or might use some even more complicated probability distribution), and thus deniability becomes harder to argue. Precise Zero Knowledge instead stipulates essentially optimal deniability: whatever view the gate keeper obtains with our help, he could have generated alone in twice its running time in that view.

Results Without any trusted set-up assumptions, none of the known zero-knowledge protocols and simulators satisfy our precise simulation requirement. In particular, in Appendix 5, we show that precise zero-knowledge proofs systems with *black-box simulators* only exists for "trivial" languages. (In fact, our impossibility results is even stronger; we also rule out the possibility of proof systems

for languages in $\mathcal{BPP}$ where the running time of the honest verifier is significantly smaller than the time needed to decide the language.) Since all classical protocols and simulators are black-box this result shows that none of these simulators satisfies our precise $\mathcal{ZK}$ requirements (even for the weakest notion of computationally precise $\mathcal{ZK}$). It can furthermore be verified that also known *non-black-box* simulator techniques (due to Barak [2]) are insufficient. We provide more details on such simulations in section C.[4]

We, however, manage to prove the existence of Precise $\mathcal{ZK}$ protocols in a variety of settings under standard complexity assumptions. Namely, we prove the following.

Theorem: *Assume the existence of $k(n)$-round public-coin statistically-hiding commitments. Then, every language in $\mathcal{NP}$ has:*

1. *an $\omega(k(n))$-round computational $\mathcal{ZK}$ proof with polynomial precision.*
2. *an $\omega(k(n)\log n)$-round computational $\mathcal{ZK}$ proof with linear precision.*
3. *an $k(n)+\omega(1)$-round statistical $\mathcal{ZK}$ argument with polynomial precision.*
4. *an $k(n)+\omega(\log n)$-round statistical $\mathcal{ZK}$ argument with linear precision.*

Furthermore, every language in $\mathcal{IP}$ has a computational $\mathcal{ZK}$ proof with linear precision.

Theorem: *Assume the existence of one-way functions. Then there exist an $\omega(1)$-round computational $\mathcal{ZK}$ argument with polynomial precision for all languages in $\mathcal{NP}$. There also exists an $\omega(\log n)$-round computational $\mathcal{ZK}$ argument with linear precision for all languages in $\mathcal{NP}$.*

To prove the above result, while avoiding our impossibility ones, we rely on a "slightly" non-black-box simulation technique: the only non-black-box components of it is that we allow the simulator to *count* the number of computational steps taken by a verifier, and *times out* the verifier when it runs for too long.

We also prove that statistically precise $\mathcal{ZK}$ proofs exist unconditionally for some notable non-trivial languages. In particular,

Theorem: *There exist an $\omega(1)$-round statistical $\mathcal{ZK}$ proofs with polynomial precision for Graph Non-isomorphism and Quadratic Non-residuosity. There also exist $\omega(\log n)$ rounds statistical $\mathcal{ZK}$ proofs with linear precision for Graph Non-isomorphism and Quadratic Non-residuosity.*

The last result can be generalized to provide statistically precise $\mathcal{ZK}$ proofs for a *restricted* version of the Statistical Difference problem, $\mathbf{SD}^1_{1/2}$ [69, 57]. (The general, i.e., non-restricted, Statistical Difference problem is complete for statistical $\mathcal{ZK}$ [69])

[4] Very briefly, the reason why the non black-box simulator of Barak [2] (and variants thereof) is not precise arises from the fact that the simulator will *always* commit to the whole auxiliary tape of the verifier (which might be very long), while the verifier with high probability might read only a very small portion of it. That is, the running time of the simulator will always be "large", while the verifier might run very fast with high probability.

An outline of our techniques. On a high-level, our protocols follows a paradigm by Feige and Shamir [28]. The protocols consists of two phases: the first phase is a *pre-amble* phase, the second phase is a *proof* phase. The pre-amble phase has the property that the prover's messages can be perfectly emulated (without knowing the witness). Additionally, during this pre-amble phase, a simulator will be able to, by "rewinding" the verifier, recover a "trapdoor" that it can later use in the proof phase to provide a convincing (and indistinguishable) proof without knowing the witness. The traditional way of simulating such a protocol is to first honestly emulate the pre-amble phase, and next rewind the verifier (in the pre-amble phase) to "extract" out a trapdoor that can later be used in the proof phase. The problem with using this approach in the context of Precise $\mathcal{ZK}$ is that the verifier may run "fast" in the first execution of the pre-amble phase, but run "long" in the rewindings. This prevents the simulator from outputting the view of the first execution of the pre-amble phase; rather, to make sure that the simulation respects the running-time of the verifier in the view output, it would need to output a view of the pre-amble plase where the verifier ran "long". But if we do this, we biase the views outputs towards being "long".

To get around this problem, we will "measure" how many computational steps the verifier takes in the first execution of the pre-amble phase, and next, in the rewindings, "cut-off" the verifier if it attemps to take more steps than it did in the first run. Doing this, however, introduces an additional problem: now, we can no longer guarantee that the number of rewindings needed to perform the extraction is small. We solve this issue by increasing the length of the pre-amble phase: following the work of Richardson and Kilian [66], (see also [50, 65]), we consider a pre-amble phase which gives the simulator many sequential "rewinding" opportunities from which a trapdoor can be extracted; we refer to these rewinding opportunities as "slots". Using a counting argument, we can next show that if the number of slots is large enough, there exists at least one of these, for which a bounded number of "rewinds-with-cutoff" suffices. Roughly speaking, the idea is that for every slot s, the probability that the verifier runs longer in the first execution of the slot s than in the "rewinding" of s, is the same as the probability that it runs shorter in the first execution of s than in the rewinding, and thus this probability must be $\frac{1}{2}$. If we instantiate the pre-amble with a protocol where each slot has the property that a trapdoor can be extracted using a *single* successful rewinding, then for each slot in the pre-amble, we have a probability of $\frac{1}{2}$ of succeeding in extracting a trapdoor from this slot, using just a single "rewiding-with-cutoff". This concludes that if we have $\omega(\log n)$ sequential slots, we can ensure that, except with negligible probability, a trapdoor is always extracted from the pre-amble phase. Furthemore, if we rely on this method for extracting the trapdoor from the pre-amble phase, we can guarantee that if the verifier took t steps in the first run of the pre-amble phase, then the extraction process will take at most t steps (since we cut-off the verifier if he runs longer in the rewindings than what he did in the first run). Using this method, we can demonstrate a $\omega(\log n)$ round protocol with precision $p(n,t) = 2t$.

Other complexity measures. In this paper we focus only on precision with respect to the complexity measure *running time*. Our notions can, however, be instantiated also with other complexity measures. A treatment of precision with general complexity measures can be found in [47].

1.2 Precision Beyond $\mathcal{ZK}$

We extend the treatment of precision also to other cryptographic primitives.

Precise Proofs of Knowledge The notion of a proof of knowledge was intuitively introduced by Goldwasser, Micali and Rackoff [37] and was formalized by Feige, Fiat and Shamir [26], Tompa and Woll [71] and Bellare and Goldreich [5]. Loosely speaking, an interactive proof of $x \in L$ is a proof of knowledge if the prover can only convinces the verifier (with non-negligible probability), if it in PPT can compute a witness w for the statement x. This is formalized by requiring the existence of a PPT machine E, called an *extractor*, such that E on input the description of P and any statement $x \in L$ outputs a valid witness for $x \in L$ if P succeeds in convincing the Verifier (with non-negligible probability) that $x \in L$. Halevi and Micali [44] introduced a strengthening of the notion of a proof of knowledge, called *a conservative proof of knowledge*, which guarantees a tighter coupling between the expected running-time of the extractor E and the *expected running time of P in an interaction with the honest verifier*. Their notion thus guarantees that P will only be able to convince the Verifier that $x \in L$, if it could have computed a witness for $x \in L$ in time closely related to its expected running-time.

The above notions were all designed to be applicable within cryptographic protocols (e.g., as tool for achieving $\mathcal{ZK}$, or for identification protocols). Thus, in a sense, these definitions are syntactic rather than semantic. We believe that is it important to have a semantical notion of a proof of knowledge, which corresponds as closely as possibly to the "intuitive" meaning of what it means to verify an agent's knowledge. For instance, such a notion could conceivably be applicable to provide a formal treatment of, say, school exams.

Towards this goal, we put forward a notion of a proof of knowledge that more precisely bounds the knowledge of the prover in an *execution by execution* manner: Consider, for instance, a teacher wishing to verify whether a student has done its homework. We desire a test that checks if the student indeed did its homework (and knows what it is supposed to know), and not merely that the student *could* have done it (under some different circumstances).[5] Thus, we would like to have a notion of a proof of knowledge which requires that a prover (the student) can only convince the verifier (the teacher) whenever it "essentially" has a witness on its worktape (its brain) at the end of the interaction (the exam). More precisely, we put forward a notion of a proof of knowledge which guarantees that

> *P will only succeed in convincing the verifier that $x \in L$ if P can compute a witness for $x \in L$ in time closely related (say, within a constant factor) to the **actual** time P spent in the every interaction where V is accepting.*

That is, whenever P spends t steps *in a particular interaction* in order to convince the Verifier, P could in, say, $2t$, steps compute a witness to the statement proved. As such, our definition allows us to capture what it means for a particular prover to know a witness in a *particular* interaction, providing more intrinsic meaning to the notion of a proof of knowledge. We call a proof satisfying the above property a *precise proof of knowledge*.

Just as traditional proofs of knowledge protocols are useful in the design of zero-knowledge proofs (and more general secure protocol), we demonstrate the applicability of precise proofs of knowledge protocols as building blocks in order to obtain precise zero-knowledge proofs. (In fact, we here show that a slightly weaker variant of our notion of precise proofs of knowledge, called *emulatable precise proofs of knowledge*, is sufficient; roughly speaking, this notion is a precise analog of the notion of witness extended emulation of Lindell [52]).

[5]The student might e.g., decide to do the homework based on what questions the teacher asks. In this case, we only want the teacher to accept in the event that the student indeed did the homework.

Precise Encryption We provide a strengthening of the traditional semantical security definition of [36]. In fact, our strengthening applies also to Shannon's original definition of prefect secrecy—as far as we know, this is the first strengthening of Shannon's definition (when considering single-message security under a passive attack). Intuitively, (for the case of public-key encryption)

> *We say that an encryption scheme is secure with precision $p(n,t)$ if anything an eavesdropper learns in time t, given the encryption of a message of length n, could have been computed in time $p(n,t)$, knowing only n and the public-key.*

Recall that the definition of [36] only requires that the eavesdropper learns no more than what could have been computed in polynomial time. Due to the equivalence between semantical-security and indistinguishability [36], it is, however, easy to see that any semantically-secure encryption scheme has precision $p(n,t) = O(t + g(n))$ where g is a polynomial upper-bounding the running-time of the encryption algorithm.[6] Thus, semantically-secure encryption schemes are already "quite" precise (in contrast to traditional $\mathcal{ZK}$).

Nonetheless, we argue that it is still be desirable to improve the precision (both for philosophical and practical purposes). Consider, for instance, an adversary with small computational resources (e.g, a constant-depth circuit). It conceivable that such an adversary cannot execute the encryption procedure (in particular if the communicating parties are powerful and paranoid entities, and as a consequence use a very large security parameter), yet we would still like to guarantee that seeing an encrypted messages does not provide the adversary with additional knowledge.

We investigate whether better precision can be obtained. The answer is surprisingly elegant:

> *Any semantically-secure encryption scheme with pseudorandom ciphertexts has* linear *precision—i.e., precision $p(n,t) = O(t)$.*

This result (although technically simple) gives a strong semantical motivation for the "lay-man" belief that a good encryption scheme should scramble a message into something random (looking)—indeed, as demonstrated, whenever this happens the encryption scheme satisfies a semantically stronger notion. (Additionally, the same argument shows that the one-time pad is perfectly secure with essentially optimal precision; however, not necessarily every perfectly secure encryption scheme has linear precision.)

We additionally show how to turn any CCA2-secure encryption scheme into one that is CCA2-secure with linear precision; the transformation (which relies on a careful padding argument) is only a feasibility results (and does not preserve the efficiency of the underlying CCA2-secure scheme more than up a polynomial factor). We leave open the question of constructing practical CCA2-secure precise encryption schemes.

Precise Secure Computation We finally provide a definition of precise secure computation. Whereas the definitions of precise encryption and precise proofs of knowledge are quite different from the traditional definitions, the definition of precision secure computation is instead "just" a careful adaptation of the precise $\mathcal{ZK}$ definition to the setting of secure computation. Additionally, we show that by appropriately compiling the GMW protocol [34] using precise $\mathcal{ZK}$ protocols (and applying careful padding), results in a secure computation protocol with linear precision. Similar modifications can be done also to the protocols of [8] and [68].

[6] This follows since the view of an adversary can be readily simulated by honestly encrypting 0^n.

1.3 Related Work

Our notions have benefited from several prior ones, in particular knowledge tightness for zero-knowledge proofs [34, 31]. It should also be appreciated that the on-line simulators of non-interactive zero-knowledge proofs [12, 14] as well as the on-line simulators for secure computation of [58, 21, 15] are specific, earlier examples of precise simulators. However, such on-line simulators require trusted set-up assumptions, or an honest majority, or super-polynomial simulation [63]. (In a sense, therefore, we show that *for pure precision purposes* none of those requirements are necessary.) The basic deficiencies of expected-polynomial-time simulators with respect to strict polynomial-time ones pointed out in [7] have motivated our work. As discussed later, our definitions capture many of the desiderata of [44] for proofs of knowledge. We also mention that Yuval Ishai independently has advocated an execution-by-execution preservance of resources (such as corruption of players) in the context of secure computation.

1.4 Subsequent Work

Several recent work extend our work. We highligth three directions below.

Precision and Game/Decision Theory Halpern and Pass [47] provide a model of "game-theory with costly computation"—where the players are charged for the computational complexity of their strategies. As they show, in this model, the traditional game-theoretic notion of *Nash-implementation of mediators* is closely related to the notion of Precise Secure Computation. Thus, in a sense, precision allows us to relate the zero-knowledge simulation paradigm and Nash equilibrium.

Halpern and Pass [48] also present a connection between Precise $\mathcal{ZK}$ and the decision-theoretic concept of *value of information*: roughly speaking, in games with costly computation, the value of participating in a $\mathcal{ZK}$ proof with precision $p(n,t) = 2t$ is no higher than the value of getting a twice as fast computer.

Precision and Concurrency Pandey, Pass, Sahai, Teng and Venkitasubramariam [62] extend our treatment to the concurrent setting, where we consider the execution of multiple zero-knowledge protocols taking place at the same time [23]; in particular, they present Precise Concurrent $\mathcal{ZK}$ protocols for $\mathcal{NP}$.

A very recent work by Goyal, Jain and Ostrovsky [40] present a surprising application of Precise $\mathcal{ZK}$ to the construction of concurrently-secure *password key-exchange protocols*.

Precision and Leakage-resilient Protocol A very recent work by Garg, Jain and Sahai [41] present an elegant application of Precise $\mathcal{ZK}$ to the construction of leakage-resilient protocol—that is, cryptographic protocol that retain their security properties, even if part of the state of the honest parties is leaked to the attacker. Roughly speaking (and oversimplifying), the idea is to construct a precise $\mathcal{ZK}$ protocol with respect to the complexity measure "leakage".

1.5 Notation and Preliminaries

Our probabilistic notation follows [39]; see appendix A.1 for more details. By algorithm, we mean a Turing machine. For simplicity, we only consider algorithms with finite (i.e., bounded) running-time. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, the notation "$M_r(x)$" denotes the

output of the M on input x when receiving r as random tape. We let the notation "$M_\bullet(x)$" denote the probability distribution over the outputs of M on input x where each bit of the random tape r is selected at random and independently, and then outputting $M_r(x)$ (note that this is a well-defined probability distribution since we only consider algorithms with finite running-time). Our protocol notation is described in Appendix A.2. We emphasize that in our notion of a *view* of an interaction, we only consider the part of the input and random tapes *actually* read by the parties.

When measuring the running-time of algorithms, for simplicity (and in accordance with the literature, see e.g., [31]), we assume that an algorithm M, given the code of a second algorithm A and an input x, can emulate the computation of A on input x with no (or linear) overhead.

Other preliminaries, such as the definition of indistinguishability of ensembles, interactive proofs, zero-knowledge proofs, witness indistinguishability, proofs of knowledge and commitments can be found in Appendix B.

1.6 Overview

In Section 2 we provide formal definitions of precise $\mathcal{ZK}$ and precise proofs of knowledge. Our constructions of precise (statistical or computational) $\mathcal{ZK}$ (proof/argument) protocols (with polynomial or linear precision) essentially proceed in two steps: first, we construct a witness-indistinguishable ($\mathcal{WI}$) [27] precise proof of knowledge, and then use it to yield a corresponding precise $\mathcal{ZK}$ protocol. All main technical difficulties arise in the first step. The second one can sometimes be obtained as easily as by replacing a standard proof of knowledge, in a prior $\mathcal{ZK}$ construction, with our precise one. We actually obtain our precise proofs of knowledge in an essentially uniform way. We start by showing some key *knowledge precision lemmas*. Each such lemma shows that a $\mathcal{WI}$ proof of knowledge (P, V), when repeated a sufficient number of times m, yields a $\mathcal{WI}$ *precise* proof of knowledge, as long as (P, V) satisfies *special soundness* [19].[7] (Different lemmas state that different number of repetitions yield different levels of precision—the higher the number the better the precision.) The knowledge precision lemmas can be found in Section 3. In Section 4, we rely on the constructed proof of knowledge protocols to provide constructions of precise $\mathcal{ZK}$ protocols.

Section 5 contains our black-box lower bounds for precise $\mathcal{ZK}$.

In Section 6 and Section 7, we extend the notion of precision to secure encryption and secure computation.

2 Definitions of Precise Zero Knowledge and Proofs of Knowledge

In this chapter we put forward our precise notions of zero knowledge and proofs of knowledge. We also investigates some basic properties of our new notions.

2.1 Precise Zero Knowledge

Definition 1 (Perfect Precise $\mathcal{ZK}$) *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P, V) an interactive proof (argument) system for L, and $p : N \times N \times N \to N$ a monotonically increasing function. We say that (P, V) is* perfect $\mathcal{ZK}$ *with* precision p *if, for every ITM V', there exists a probabilistic algorithm S such that the following two conditions holds:*

1. *The following two ensembles are identical:*

[7] Despite the name, the quite standard property that a valid witness can be readily computed from any two executions having the same first message but different second messages.

(a) $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

(b) $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

2. *For every* $x \in L$, *every* $z \in \{0,1\}^*$, *and every sufficiently long* $r \in \{0,1\}^*$, $\text{STEPS}_{S_r(x,z)} \leq p(|x|, \text{STEPS}_{V'}(S_r(x,z)))$.

We refer to an algorithm S as above as a *precise simulator*, or as a *simulator with precision* p. If $p(n,t)$ is a polynomial (a linear function) in *only* t , we say that (P,V) has polynomial (linear) precision.

Computational/Statistical $\mathcal{ZK}$. We obtain the notion of *statistically precise* $\mathcal{ZK}$ by requiring that the two ensembles of Condition 1 be statistically close over L. We obtain the notion of a *computationally precise* $\mathcal{ZK}$ by furthermore adding the restriction that V' is a probabilistic polynomial-time machine, and by requiring that the two ensembles of Condition 1 are computationally indistinguishable over L.

Remarks:

1. Note that in the case of computationally precise $\mathcal{ZK}$ our definition only differs from the standard definition of $\mathcal{ZK}$ in that we additionally require that the actual running-time of the simulator is "close" to the actual running-time of the verifier in a true interactions with a prover.

   In the case of perfectly and statistically precise $\mathcal{ZK}$, our definition additionally differs in that we require simulation of *all* malicious verifiers (even those having an unbounded running time). By contrast, perfect/statistical $\mathcal{ZK}$ in the standard sense only calls for polynomial-time verifiers to be simulatable (though, for classical examples of perfect/statistical $\mathcal{ZK}$ proofs, all verifiers can actually be simulated).[8]

2. Note that every perfect/statistical/computational $\mathcal{ZK}$ proof system (P,V) with polynomial precision is a perfect/statistical/computational $\mathcal{ZK}$ proof system in the standard sense.[9] The converse, however, may *not* be true—for every fixed polynomial p there might exist a verifier having a worst-case (or even expected) runnning-time that exceeds p by far, but whose actual running time t is small a substantial amount of the time. Consider, for instance, a verifier V' that with probability $\frac{1}{100}$ takes n^{50} steps, and otherwise n steps. The expected running-time of V' is thus $O(n^{50})$; we conclude that even a simulator with optimal expected precision could potentially *always* take $O(n^{50})$ steps, whereas V' almost always takes n steps. In fact, even a simulator with optimal higher-moment precision might always take $O(n^{50})$ steps.

3. One might consider weakening Condition 2 by requiring that it holds only for most (rather than all) tapes r of S. However, under such a relaxation, perfect $\mathcal{ZK}$ with polynomial precision would no longer imply perfect (nor statistical, or computational!) $\mathcal{ZK}$ in the standard sense. Consider a simulator S that runs in fixed polynomial time, except for a fraction $2^{-|x|}$ of its random tapes, where it always takes $2^{|x|^2}$ steps. Such an S would run in expected exponential time, but still satisfy polynomial precision under the above relaxation of Condition 2.

[8] This is not a new observation. Indeed, the definition of statistical *black-box* zero-knowledge of [72] calls for simulation of also unbounded verifiers.

[9] Indeed, even with respect to strict polynomial time simulators.

4. A seemingly stronger definition would be to require the existence of a *universal* precise simulator which works for all verifiers (assuming that it also gets the code of the verifier). Interestingly, in the case of perfectly and statistically precise $\mathcal{ZK}$ this alternative formulation is equivalent to our definition. This follows from the fact that it is sufficient to describe a precise simulator for the Universal Turing Machine that runs the code it receives as its auxiliary input. Note that the same argument does not go through for computationally precise $\mathcal{ZK}$ (nor the standard notion of $\mathcal{ZK}$): as the Universal Turing Machine is not a polynomial-time machine, computationally precise $\mathcal{ZK}$ (or standard $\mathcal{ZK}$) does not require the existence of a simulator for it.

2.2 Properties of Precise ZK

2.2.1 Preserving Running-time Distribution

Whereas the notion of knowledge-tight $\mathcal{ZK}$ (see Definition B.4) guarantees that the (expected) running-time of the simulator is closely related to the *worst-case* running-time of the adversarial verifier, we here show that the notion of precise $\mathcal{ZK}$ guarantees that the actual running-time distribution of the verifier is respected by the simulator.

We proceed to a formal treatment. The following proposition shows that the cumulative probability distribution function (cdf) of the running-time of the simulator respects the cdf of the running-time of the adversary verifier.

Proposition 1 *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, $p : N \times N \to N$ a monotonically increasing function, and (P, V) a statistical (computational resp.) $\mathcal{ZK}$ argument system for L with precision p. Let V' be an arbitrary (polynomial-time resp.) probabilistic machine and let S be the precise simulator for V'. Then there exists a negligible function $\mu(n)$, such that for all $x \in L$, $y \in R_L(x)$ all $z \in \{0,1\}^*$, it holds that for every $t \in N$:*

$$F_{V'}(t) \leq F_S(p(|x|, t)) + \mu(|x|)$$

where

$$F_{V'}(t) = \Pr\left[v \leftarrow \text{VIEW}_2[P_\bullet(x, y) \leftrightarrow V'_\bullet(x, z)] \ : \ \text{STEPS}_{V'}(v) \leq t\right]$$

and

$$F_S(t) = \Pr\left[\text{STEPS}_{S_\bullet(x,z)} \leq t\right]$$

Proof: Suppose for contradiction that there exists a (polynomial-time in the case of computational $\mathcal{ZK}$, and arbitrary otherwise) verifier V' and a polynomial $g(n)$ such that for infinitely many $x \in L$ there exists $y \in R_L(x), z \in \{0,1\}^*, t \in N$ such that:

$$F_{V'}(t) \geq F_S(p(|x|, t)) + \frac{1}{g(|x|)}$$

where S is a precise simulator for V'. Towards the goal of contradicting the precise simulation requirement of S, consider a generic x, y, z, t for which this happens. Consider the distinguisher D defined as follows:

- D on input a view v outputs 1 if and only if $\text{STEPS}_{V'}(v) \leq \text{TIME}_{V'}(|x|)$ and $\text{STEPS}_{V'}(v) \leq t$.

First, note that if V' is polynomial-time, then so is D. It follows directly from the construction of D that D on input a random view $v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]$ output 1 with probability $F_{V'}(t)$. Secondly, if follows from the precise "reconstruction" requirement of S (i.e., that the actual number of steps used by S to output a view is at most $p(|x|,t)$ where t is the running-time of V' in the view output by S) that D on input a random view $v \leftarrow S_\bullet(x,z)$ outputs 1 with probability *smaller or equal to* $F_S(p(|x|,t))$. We conclude that D distinguishes the output of $S_\bullet(x,z)$ from the view of $V'(x,z)$ in a real execution with $P(x,y)$, with probability at least $\frac{1}{g(|x|)}$, which contradicts the fact that S is a valid simulator for V'. ■

By using exactly the same proof we also get:

Proposition 2 *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, $p : N \times N \rightarrow N$ a monotonically increasing function, and (P,V) a statistical (computational resp.) $\mathcal{ZK}$ argument system for L with precision p. Let V' be an arbitrary (polynomial-time resp.) probabilistic machine and let S be the precise simulator for V'. Then there exists a negligible function $\mu(n)$, such that for all $x \in L$, $y \in R_L(x)$ all $z \in \{0,1\}^*$, it holds that for every $t \in N$:*

$$\bar{F}_{V'}(t) \geq \bar{F}_S(p(|x|,t)) + \mu(|x|)$$

where

$$\bar{F}_{V'}(t) = \Pr\left[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)] \ : \ \text{STEPS}_{V'}(v) \geq t\right]$$

and

$$\bar{F}_S(t) = \Pr\left[\text{STEPS}_{S_\bullet(x,z)} \geq t\right]$$

2.2.2 Composition of Precise $\mathcal{ZK}$

Sequential Composition. Whereas the standard definition of $\mathcal{ZK}$ only talks about a *single* execution between a prover and a verifier, Goldreich and Oren [35] have shown that the standard definition of $\mathcal{ZK}$ (see Definition 13) in fact is closed under sequential composition. That is, sequential repetitions of a $\mathcal{ZK}$ protocol results in a new protocol that still remains $\mathcal{ZK}$.

We observe that the exactly the same proof as was used by Goldreich and Oren [35] can by used to show that the protocol resulting from sequentially repeating a precise $\mathcal{ZK}$ also is a precise $\mathcal{ZK}$ proof (albeit with slightly worse precision).

Lemma 1 (Sequential Composition Theorem) *Let (P,V) be a perfect/statistical/ computational $\mathcal{ZK}$ with precision p interactive proof (or argument) for the language $L \in \mathcal{NP}$. Let $Q(n)$ be a polynomial, and let (P_Q, V_Q) be an interactive proof (argument) that on common input $x \in \{0,1\}^n$ proceeds in $Q(n)$ phases, each on them consisting of an execution of the interactive proof (P,V) on common input x (each time with independent random coins). Then (P_Q, V_Q) is an perfect/statistical/computational $\mathcal{ZK}$ interactive proof (argument) with precision $p_Q(n,t) = O(Q(n)p(n,t))$*

Parallel Composition. Goldreich and Krawczyk [33] (see also Feige and Shamir [28]) show that the standard notion of $\mathcal{ZK}$ is not closed under *parallel* repetitions. More precisely, they show that there exists $\mathcal{ZK}$ proofs, that have the property that a malicious verifier participating in 2 parallel (i.e., simultaneous and synchronized) executions of the same protocol in fact can recover the whole witness to the statement proved.

We observe that the protocol of Feige and Shamir, when instantiated with the corresponding precise $\mathcal{ZK}$ protocols (as we construct in Section 4) suffices to show that also the notion of precise $\mathcal{ZK}$ is not closed under parallel repetition.

2.3 Precise Proofs of Knowledge

We define our notion of a precise proof of knowledge. Intuitively we say that (P, V) is a proof of knowledge with precision p, if there for every adversary prover P' exists an extractor E such that:

1. Given any joint-view $(view_{P'}, view_V)$ of an execution between P' and V on common input x, it holds that E on input only the view $view_{P'}$ outputs a valid witness for $x \in L$, if $view_V$ is a view where V is accepting.

2. Given any view $view_{P'}$ containing a proof of the statement x, it furthermore holds that the *worst-case* running-time of E on input $view_{P'}$ is smaller than $p(|x|, t)$ where t denotes the *actual* running-time of P' in the view $view_{P'}$.

More precisely,

Definition 2 (Precise Proof of Knowledge) *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P, V) an interactive proof (argument) system for L, and $p : N \times N \to N$ a monotonically increasing function. We say that (P, V) is a* perfectly-sound proof of knowledge *with* precision p *for the witness relation R_L, if for every probabilistic interactive machine P', there exists a probabilistic algorithm E such that the following conditions hold:*

1. *For every $x, z \in \{0,1\}^*$,*

$$\Pr\Big[(view_{P'}, view_V) \leftarrow P'_\bullet(x, z) \leftrightarrow V_\bullet(x) : \\ \text{OUT}_V(view_V) = 1 \wedge E(view_{P'}) \notin R_L(x)\Big] = 0$$

2. *For every view $view_{P'}$ which contains the view of a proof of the statement x and every sufficiently long $r \in \{0,1\}^*$ it holds that*

$$\text{STEPS}_{E_r(view_{P'})} \leq p(|x|, \text{STEPS}_{P'}(view_{P'}))$$

We refer to an algorithm E as above as a *precise extractor*, or as an *extractor with precision p*.

Statistically/Computationally sound precise proofs of knowledge. As in the case of statistically-sound expected proofs of knowledge, we obtain the notion of a *statistically-sound precise proof of knowledge* by exchanging condition 1 in Definition 2 for the following condition:

1′. There exists some negligible function $\mu(\cdot)$ such that for every $x, z \in \{0,1\}^*$,

$$\Pr\Big[(view_{P'}, view_V) \leftarrow P'_\bullet(x, z) \leftrightarrow V_\bullet(x) : \\ \text{OUT}_V(view_V) = 1 \wedge E(view_{P'}) \notin R_L(x)\Big] \leq \mu(|x|)$$

We obtain the notion of a *computationally-sound precise proof of knowledge* by furthermore adding the restriction that P' is a probabilistic polynomial-time machine.

2.3.1 Emulatable Precise Proofs of Knowledge

We present a somewhat different notion of a proof of knowledge, called a emulatable precise proof of knowledge. This notion seems more suitable for many cryptographic applications (and in particular ours). As we elaborate upon in Section 2.3.2, this notion combines in a rather natural way the notions of precise $\mathcal{ZK}$ and precise proofs of knowledge.

In essence, we require that given an alleged prover P' and an input $x \in L$, (a) the *joint* view of P' and the honest verifier V on input x, and (b) a valid witness for $x \in L$ whenever V's view is accepting, can be simultaneously reconstructed in a time that is essentially identical to that taken by P' in the reconstructed view.

We mention that although the definition of an emulatable precise proofs of knowledge bears certain similarities with Lindell's definition of *witness extended emulation* [52], it differs in several crucial ways, as will be discussed shortly.

Definition 3 (Emulatable Precise Proof of Knowledge) *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P, V) an interactive proof (argument) system for L, and $p : N \times N \to N$ a monotonically increasing function. We say that (P, V) is a* perfectly-sound emulatable proof of knowledge *with* precision p *for the witness relation R_L, if for every probabilistic interactive machine P', there exists a probabilistic algorithm E such that the following conditions hold:*

1. *The following two ensembles are identical*
   (a) $\left\{P'_\bullet(x,z) \leftrightarrow V_\bullet(x)\right\}_{x,z\in\{0,1\}^*}$
   (b) $\left\{(view_{P'}, view_V, w) \leftarrow E_\bullet(x,z) : (view_{P'}, view_V)\right\}_{x,z\in\{0,1\}^*}$
2. *For every $x, z \in \{0,1\}^*$,*
$$\Pr\Big[(view_{P'}, view_V, w) \leftarrow E_\bullet(x,z) : \\ \text{OUT}_V(view_V) = 1 \wedge (x,w) \notin R_L\Big] = 0$$
3. *For every $x, z \in \{0,1\}^*$, and sufficiently long $r \in \{0,1\}^*$, given $(view_{P'}, view_V, w) = E_r(x,z)$, it holds that*
$$\text{STEPS}_{E_r(x,z)} \leq p(|x|, \text{STEPS}_{P'}(view_{P'}))$$

We refer to an algorithm E as above as a *precise emulator-extractor*, or as an *emulator-extractor with precision p*.

Statistically/Computationally sound emulatable precise proofs of knowledge. We obtain the notion of a *statistically-sound emulatable precise proof of knowledge* by exchanging condition 1 in Definition 3 for the following condition:

2′. There exists some negligible function $\mu(\cdot)$ such that for every $x, z \in \{0,1\}^*$,

$$\Pr\Big[(view_{P'}, view_V, w) \leftarrow E_\bullet(x,z) : \\ \text{OUT}_V(view_V) = 1 \wedge (x,w) \notin R_L\Big] \leq \mu(|x|)$$

We obtain the notion of a *computationally-sound emulatable precise proof of knowledge* by furthermore adding the restriction that P' is a probabilistic polynomial-time machine.

Remark: We note that, besides the *precise reconstruction* requirement (which is the principal difference), our definition differs also from that of [52] in that the latter only requires reconstruction of the view of the Verifier. By requiring reconstruction of the joint view of the Prover and the Verifier, we make it easier to handle proofs of knowledge as a sub-protocol: Whenever, in a larger protocol, algorithm A gives a proof of knowledge to B about $x \in L$, we can syntactically replace their views with the output of our extractor on inputs A and x, without altering in any way their distributions. Furthermore, our extractor will return, on the side, a valid witness for $x \in L$, whenever the view of B is accepting.

2.3.2 $\mathcal{ZK}$ for the Prover and Emulatable Proofs of Knowledge

In this section we show a natural relation between precise $\mathcal{ZK}$, precise proofs of knowledge and emulatable precise proofs of knowledge. More precisely, we present a lemma showing that precise proofs of knowledge protocols that are precise $\mathcal{ZK}$ *for the Prover* (i.e., the prover learns precisely nothing from the verifier) are emulatable precise proofs of knowledge. This lemma will turn out to be very useful to us since "natural" proofs of knowledge protocols (and in particular the ones we consider) have the property of being precise $\mathcal{ZK}$ for the prover.

Definition 4 (Precise $\mathcal{ZK}$ for the Prover) *Let (P, V) be an interactive proof (argument) system and $p : N \times N \to N$ a monotonically increasing function. We say that (P, V) is* $\mathcal{ZK}$ for the prover *with* precision p *if, for every ITM P', there exists a probabilistic algorithm S such that the following two conditions holds:*

1. *The following two ensembles are identical:*
   (a) $\left\{P_\bullet(x, z) \leftrightarrow V'_\bullet(x)\right\}_{x,z\in\{0,1\}^*}$
   (b) $\left\{S_\bullet(x, z)\right\}_{x,z\in\{0,1\}^*}$
2. *For every $x, z \in \{0, 1\}^*$, and sufficiently long $r \in \{0, 1\}^*$*

$$\text{STEPS}_{S_r(x,z)} \leq p(|x|, \text{STEPS}_{P'}(view_1[S_r(x, z)]))$$

Remarks: We point out that the definition of "precise $\mathcal{ZK}$ for the prover" differs from the definition of "precise $\mathcal{ZK}$" in that we require the simulator to output the joint view of both the prover and verifier. Note that this difference becomes insubstantial if we assume that the verifier V reveals all its random coins in the last round—then the view of V can be reconstructed from the view of the P' in time that is proportional to the running time of P'. Although this assumption is without loss of generality, we prefer to present the definition of "precise $\mathcal{ZK}$ for the prover" in its current form to emphasize the need for the simulator to output also the view of the verifier.

Lemma 2 *Let (P, V) be a perfectly-sound (statistically-sound/computationally-sound) proof of knowledge with precision $p_1(n, t)$ for the witness relation R_L. If (P, V) is $\mathcal{ZK}$ for the prover with precision $p_2(n, t)$, then (P, V) is a perfectly-sound (statistically-sound/computationally-sound) emulatable proof of knowledge with precision $O(1) \cdot [p_1(n, t) + p_2(n, t)]$ for the witness relation R_L.*

Proof: For a given prover P', consider the "precise $\mathcal{ZK}$ for the prover" simulator S, and the extractor E. We construct an emulator-extractor E_2 for P'. E_2 on input x, z' proceeds as follows:

1. $(view_{P'}, view_V) \leftarrow S_\bullet(x, z)$.
2. $w \leftarrow E_\bullet(view_{P'})$.
3. Output $((view_{P'}, view_V), w)$.

It directly follows from the validity of S and E that the output of E_2 is correctly distributed, and that its precision is $O(1) \cdot [p_1(n,t) + p_2(n,t)]$. ■

3 Constructions of Precise Proofs of Knowledge

We provide four different "knowledge precision" lemmas showing how to transform standard proof of knowledge protocols with certain specific features into precise proofs of knowledge. All our transformations are very simple and follow the same paradigm: an "atomic" standard proof of knowledge (with specific properties) is repeated sequentially an appropriate number of time.

More precisely, we show that,

1. $\omega(1)$ sequential repetitions of, so called, *special-sound* proofs of knowledge constitute statistically-sound proofs of knowledge with *polynomial* precision.
2. $\omega(\log n)$ sequential repetitions of, so called, *special-sound proofs of knowledge with linear extraction* constitute statistically-sound proofs of knowledge with *linear* precision.
3. $\omega(1)$ sequential repetitions of, so called, *computationally special-sound* proofs of knowledge constitute computationally-sound proofs of knowledge with *polynomial* precision.
4. $\omega(\log n)$ sequential repetitions of, so called, *computationally special-sound proofs of knowledge with linear extraction* constitute computationally-sound proofs of knowledge with *linear* precision.

We furthermore show that $\mathcal{WI}$ of the underlying atomic proof of knowledge protocols is preserved under all of the above transformations (this follows easily since the transformation only consists of sequential repetition of the underlying protocol and since $\mathcal{WI}$ is closed under composition [27]).

Before we state our lemmas we start by formally specifying what we mean by "sequential repetition". Given a function m, we say that the m-sequential repetition of (P, V) is an interactive proof system $(\tilde{P}, \tilde{V})$ defined as follows: on common input $x \in \{0,1\}^n$, $(\tilde{P}, \tilde{V})$ proceeds in $m(n)$ phases, each on them consisting of an execution of the interactive proof (P, V) on common input x (each time with independent random coins). V' finally accepts if V accepted in all $m(n)$ executions of (P, V).

3.1 Statistical Knowledge Precision Lemmas

We start by recalling the notion of *special-sound* proofs [19]. (Looking ahead, we mention that the protocol of Blum is known to be special-sound.) Intuitively, a three-round public-coin interactive proof is said to be special-sound, if a valid witness to the statement x can be readily computed from any two accepting proof-transcripts of x which have the same first message but different second messages. More generally, a k-round public-coin interactive proof is said to be special-sound if the $k-1$'st round is a verifier-round i (i.e., a round where the verifier is supposed to send a message) and

a valid witness to the statement x can be readily computed from any two accepting proof-transcripts of x which have the same first $k-2$ messages but different $k-1$'st message.

We proceed to a formal definition. We start by introducing some notation. Let $T_1 = (m_1^1, .., m_k^1)$, $T_2 = (m_1^2, .., m_k^2)$ be transcripts of a k-round protocol. We say that T_1 and T_2 are *consistent* if the first $k-2$ messages are the same, but the $k-1$'st message is different, i.e, $m_j^1 = m_j^2$ for $j < k-1$ and $m_{k-1}^1 \neq m_{k-1}^2$.

Let ACCEPT_V denote the predicate that on input a statement x and a k-round transcript of messages $T = m_1, m_2, .., m_k$ outputs 1 if and only if V accepts in that transcript (recall that our definition of public-coin protocols requires that the verifier determines whether to accept or not by applying a *deterministic* predicate to the transcript of all messages in the interaction.).

Definition 5 (Special-sound Proof) *Let (P, V) be a k-round public-coin interactive proof for the language $L \in \mathcal{NP}$ with witness relation R_L. We say that the protocol (P, V) is* special sound *with respect to R_L, if the $k-1$'st-round of (P, V) is a verifier-round and there exits a polynomial-time extractor machine X, such that for all $x \in L$ and all consistent transcripts T_1, T_2 it holds that if $\text{ACCEPT}_V(x, T_1) = 1$, $\text{ACCEPT}_V(x, T_1) = 1$ then $X(T_1, T_2, x) \in R_L(x)$.*

In the sequel we often employ the expression *verifier challenge* (or simply *challenge*) to denote the message sent by the verifier in the $k-1$'st round.

We will require the use of special-sound proofs for which extraction can be performed "very" efficiently. We say that (P, V) is *special-sound with linear extraction*, if the predicate ACCEPT_V can be computed in linear time (in its inputs length) and the extractor X in definition 5 has a linear running time.

Remark: Note that every special-sound proof can be turned into a special-sound proof with linear precision by "harmless" padding – the prover can always prepend a "dummy" string to its first message. Furthermore, note that this padding preserves properties such as $\mathcal{WI}$ of the original protocol.

It can be seen that all special-sound interactive proofs are proofs of knowledge [19, 20]. We here show that appropriate sequential repetition of a special-sound proof results in an precise proof of knowledge.

3.1.1 Linear precision using $\omega(\log n)$ rounds

We show that $\omega(\log n)$ sequential repetitions of a special-sound proof *with linear extraction*, yields a statistically-sound proof of knowledge with linear precision. More precisely, if assuming that a Turing machine can emulate another Turing machine at no cost, then this extraction will take at most $2t + poly(n)$ steps, on input a view where the prover takes t steps.

Lemma 3 (Statistical Knowledge Precision Lemma - Linear Precision) *Let (P, V) be a special-sound proof system with linear extraction for the language L with witness relation R_L. Let $m(n) = \omega(\log n)$, and let $(\tilde{P}, \tilde{V})$ denote the m-sequential repetition of (P, V). Then $(\tilde{P}, \tilde{V})$ is a statistically-sound proof of knowledge with linear precision for the language L with witness relation R_L. If, furthermore (P, V) is (statistical/perfect) $\mathcal{WI}$, then $(\tilde{P}, \tilde{V})$ is so as well.*

Proof: Let $l = l(n)$ denote the length of the verifier challenge in an execution of (P, V) on common input $x \in \{0, 1\}^n$. We describe an extractor E that uses "almost" black-box access to the malicious

prover P'. On a high-level, E on input a view $view_{P'}$ of an execution on common input x performs the following two steps:

1. In the first step, E feeds the view $view_{P'}$ to P' while recording the number of computational steps required by P' to answer each query.

2. In the second step, E uses the running-time statistics collected in the first step to attempt extracting a witness. This is done by rewinding P' *a fixed number of times* for each verifier challenge (in fact once will be sufficient), but in each rewinding cutting the execution of P' whenever P' exceeds the *actual* number of computational steps used by P' to answer the same challenge in the view $view_{P'}$.

Note that both of the above steps require a non-black box use of P' (even if in a quite minimal sense). In particular, we use the code of P' to learn the number of computational steps that P' uses to answer each challenge.

We proceed to a more formal description of E. E proceeds as follows on input a view $view_{P'}$ of an execution on common input x.

1. E checks (by applying the predicate ACCEPT) if the verifier V rejects any of the m proofs in the view $view'_P$. If so, it halts outputting $\perp$.

2. Let $(r_1, r_2, .., r_m)$ denote the verifier challenges in each of the m sequential repetitions of the atomic protocol (P, V) in the $view_{P'}$. E starts by feeding the view $view_{P'}$ to P', while at the same time keeping track of the number of computational steps that P' requires to answer each challenge r_i. Let t_i denote the number of computational steps used by P' to answer the i'th challenge (i.e., the challenge r_i of the i'th atomic protocol)

3. For each repetition $i \in \{1, .., m\}$ of the atomic protocol, E performs the following extraction procedure.

   (a) E rewinds P' to the point where the i'th challenge is supposed to be sent. (This results in the same state as if restarting P' and feeding it the $view_{P'}$ up until the message r_i is supposed to be sent.)

   (b) E feeds P' a new truly random challenge $r'_i \stackrel{R}{\leftarrow} \{0,1\}^l$, and lets P' run for at most t_i steps to compute an answer.

   (c) If an accepting answer has been obtained within t_i steps, and if the new challenge r'_i is different from r_i, E computes a witness w by applying the special-soundness extractor X to the two obtained accepting transcripts (since now two consistent and accepting transcripts of the atomic protocol (P, V), have been obtained) and halts outputting w.

4. If the extraction did not succeed in any of the m repetitions of the atomic protocol, E outputs $\perp$.

Running time of E. Let t_i denote the number of computational steps required by P' to provide an answer to the challenge in the i'th atomic protocol in the view $view_{P'}$. Furthermore, let t denote the total running time of P' in the same view. Since for each atomic protocol i, E only rewinds P'

once, and this time cuts the execution after t_i steps, it follows that attempted extraction from all m atomic protocols requires running P' for at most

$$t + \sum_{i=1}^{m} t_i \leq 2t$$

steps. Since we assume that emulation of a Turing Machine by another Turing Machine can be done with only linear overhead, and since (by the special-soundness with linear extraction property) both checking if a transcript is accepting, and extracting a witness from two accepting transcripts, can be done it time proportional to the length of the transcript, we conclude that the running time of E is a linear function of t.

Success probability of E. We show that the probability that the extraction procedure fails, on input a uniformly chosen view of P', $view_{P'}$, of an execution between $P'(z)$ and V on common input $x \in \{0,1\}^n$, is a negligible function in n, for any $z \in \{0,1\}^*$.

Towards this goal, we first analyze the probability that extraction fails for a singe instance of the atomic protocol. We assume without loss of generality that P' is a deterministic machine (this is w.l.o.g. since P' could always obtain its random tape as part of its auxiliary input z).

Consider any $i \in [m]$. We start by introducing some notation:

1. Given any view $view_{P'}$, let $view^i_{P'}$ denote the prefix of the view up until the point where P' is about to receive its i'th challenge.

2. Given any view $view_{P'}$, let $steps(view^i_{P'}, a)$ denote the number of steps P' takes to *correctly* answer the i'th challenge, when first feed the view $view^i_{P'}$, and then the i'th challenge a; if the answer by P' is incorrect (i.e., if the proof is rejecting) we set $steps(view^i_{P'}, a) = \infty$

Note that extraction (by E, on input a view $view_{P'}$) from the i'th atomic protocol only fails if either of the following happens, letting r_i denote the i'th challenge in $view_{P'}$, and r'_i the new challenge sent by E:

1. $r_i = r'_i$

2. $steps(view^i_{P'}, r_i) < steps(view^i_{P'}, r'_i)$

We start by noting that only for a fraction

$$\frac{2^l}{2^{2l}} = 2^{-l}$$

of challenges $r_i, r'_i \in \{0,1\}^l$, it holds that $r_i = r'_i$. Secondly, note that for any pair of challenges $a, b \in \{0,1\}^l$, $a \neq b$ and any prefix view v it holds that if extraction fails when $r_i = a, r'_i = b$, and $view^i_{P'} = v$, then extraction will succeed when $r_i = b, r'_i = a$, and $view^i_{P'} = v$. Thus, any pair of challenges (a, b) has a "companion" pair (b, a) such that at most one of the pairs will result in a failed extraction. Furthermore, any two pairs (a, b), (c, d) that are not each others companion, have *disjoint* companion pairs. We conclude that for any prefix view v, the number of pairs $(a, b) \in \{0,1\}^{2l}$, such that extraction succeeds if $r_i = a$, $r'_i = b$ and $view^i_{P'} = v$ is

$$\frac{2^{2l} - 2^l}{2}$$

It thus holds that the fraction of pairs $(a,b) \in \{0,1\}^{2l}$, such extraction fails if $r_i = a$, $r'_i = b$ and $view^i_{P'} = v$ is

$$\frac{2^{2l} - (2^{2l} - 2^l)/2}{2^{2l}} = \frac{1}{2} - 2^{-l-1}$$

Since for any two pairs $(a,b),(c,d) \in \{0,1\}^{2l}$, and any prefix view v, the probability (over a random input view $view_{P'}$ and the internal random coins of E) that $r_i = a$, $r'_i = b$ and $view^i_{P'} = v$ (where r_i denotes the i'th challenge in $view_{P'}$, and r'_i the new challenge sent by E) is the same as the probability that $r_i = c, r'_i = d$, and $view^i_{P'} = v$ we have that the probability that extraction fails from the i'th executions is

$$\frac{1}{2} + 2^{-l-1}$$

We proceed to show that the overall failure probability of attempted extraction from all executions $i \in [m]$ is negligible. Note that by the definition of $(\tilde{P}, \tilde{V})$ it holds that the distribution of the i'th challenge r_i sent by $\tilde{V}$ is independent of the messages sent by $\tilde{V}$ in prior executions. It also holds that the distribution of the new challenge r'_i sent by E is independent of all previously sent challenges, as well as the view it receives. We conclude that the failure probability for any execution i is independent of the failure probability of all other executions. Thus, the overall failure probability is bounded by

$$\left(\frac{1}{2} - 2^{-l-1}\right)^m \leq \left(\frac{3}{4}\right)^{\omega(\log n)}$$

Witness Indistinguishability. It follows directly from the fact that $\mathcal{WI}$ is closed under sequential composition [27] that (P', V') is $\mathcal{WI}$ if (P, V) is so. ■

3.1.2 Polynomial precision using $\omega(1)$ rounds

We proceed to show that $\omega(1)$ sequential repetitions of a special-sound proof (with negligible soundness error) yields a statistically-sound proof of knowledge with precision $p(n,t)$ where p is a polynomial in both n and t. More precisely, if assuming that a Turing machine can emulate another Turing machine at no cost, then this extraction will take at most $nt + poly(n)$ steps on input a view where the prover takes t steps. If, furthermore, the special-sound proof has the property that any prover is required to take at least $|x|$ steps to make the verifier accept a proof of x (e.g., it needs to communicate at least $|x|$ bits) it additionally holds that the resulting protocol is $\mathcal{ZK}$ with polynomial precision.

Lemma 4 (Statistical Knowledge Precision Lemma - Polynomial Precision) *Let (P,V) be a special-sound proof system for the language L with witness relation R_L. Let $m(n) = \omega(1)$, and let $(\tilde{P}, \tilde{V})$ denote the m-sequential repetition of (P,V). If in the execution of (P,V) on common input x the length of the verifier challenge is $\omega(log(|x|)$, and V always rejects unless it receives less than $|x|$ bits from the prover, then $(\tilde{P}, \tilde{V})$ is a statistically-sound proof of knowledge with polynomial precision for the language L with witness relation R_L. If, furthermore (P,V) is (statistical/perfect) $\mathcal{WI}$, then $(\tilde{P}, \tilde{V})$ is so as well.*

Proof: Let $l = l(n)$ denote the length of the verifier challenge in an execution of (P,V) on common input $x \in \{0,1\}^n$. Again, we describe an extractor E that uses almost black-box access to the malicious prover P'. The extractor E proceeds in exactly the same way as the extractor in the proof of Lemma 3, with the only exception that for each repetition i of the atomic protocol (P,V), E

rewinds P' n times (instead of only once) each time feeding it a new truly random challenge $r_i^{(j)}$, for $j \in [n]$.

For convenience, we repeat a full description of E. E proceeds as follows on input a view $view_{P'}$ of an execution on common input $x \in \{0,1\}^n$.

1. E checks (by applying the predicate ACCEPT) if the verifier V rejects any of the m proofs in the view $view'_P$. If so, it halts outputting $\perp$.

2. Let $(r_1, r_2, .., r_m)$ denote the verifier challenges in each of the m sequential repetitions of the atomic protocol (P, V), in the $view_{P'}$. E starts by feeding the view $view_{P'}$ to P', while at the same time keeping track of the number of computational steps that P' requires to answer each challenge r_i. Let t_i denote the number of computational steps used by P' to answer the i'th challenge (i.e., the challenge r_i of the i'th atomic protocol)

3. For each repetition $i \in \{1, .., m\}$ of the atomic protocol, E iterates the following extraction procedure n times.

   (a) E rewinds P' to the point where the i'th challenge is supposed to be sent.

   (b) In the j'th iteration, E feeds P' a new truly random challenge $r_i^{(j)} \xleftarrow{R} \{0,1\}^l$, and lets P' run for at most t_i steps to compute an answer.

   (c) If an accepting answer has been obtained within t_i steps, and if the new challenge $r_i^{(j)}$ is different from r_i, E computes a witness w by applying the special-soundness extractor X to the two obtained accepting transcripts (since now two consistent and accepting transcripts of the atomic protocol (P, V), have been obtained) and halts outputting w.

4. If the extraction did not succeed in any of the m repetitions of the atomic protocol, E outputs $\perp$.

Running time of E. Let t_i denote the number of computational steps required by P' to provide an answer to the challenge in the i'th atomic protocol, in the view $view_{P'}$. Furthermore, let t denote the total running time of P' in the same view. Since for each atomic protocol i, E only rewinds P' n times, and each time cuts the execution after t_i steps, it follows that attempted extraction from all m atomic protocols requires running P' for at most

$$t + \sum_{i=1}^{m} nt_i \leq (n+1)t$$

steps. Since we assume that emulation of a Turing Machine by another Turing Machine can be done with only linear overhead, and since both checking if a transcript of (P, V) is accepting, and extracting a witness from two consistent accepting transcripts of (P, V), can be done it time that is polynomial to the length of the transcript, we conclude that the running time of E is at most

$$nt + poly(n)$$

Finally, since P' must communicate at least n bits in order to convince $\tilde{V}$, we conclude that $t \geq n$, and thus the running-time of E is a polynomial function of t.

Success probability of E. We show that the probability that the extraction procedure fails, on input a uniformly chosen view of P', $view_{P'}$, of an execution between $P'(z)$ and V on common input $x \in \{0,1\}^n$, is a negligible function in n, for any $z \in \{0,1\}^*$.

As in the proof of Lemma 3 it suffices to analyze the probability that extraction fails for a singe instance of the atomic protocol. Again, we assume without loss of generality that P' is a deterministic machine.

Consider any $i \in [m]$. We use the same notation as in the proof of Lemma 3: Given any view $view_{P'}$, let $view^i_{P'}$ denote the prefix of the view up until the point where P' is about to receive its i'th challenge. Given any view $view_{P'}$, let $steps(view^i_{P'}, a)$ denote the number of steps P' takes to correctly answer the i'th challenge, when first feed the view $view^i_{P'}$, and then the i'th challenge a.

Note that if extraction (by E, on input a view $view_{P'}$) from the i'th atomic protocol fails, then either of the following events must have happened (letting r_i denote the i'th challenge in $view_{P'}$, and $r_i^{(j)}$ the j'th new challenge sent by E):

1. $r_i = r_i^{(j)}$ for some $j \in [n]$
2. $steps(view^i_{P'}, r_i) < steps(view^i_{P'}, r_i^j)$ for *all* $j \in [n]$

We start by noting that the fraction of *distinct* challenges $r_i \in \{0,1\}^l, (r_i^1, .., r_i^n) \in \{0,1\}^{nl}$, is

$$\frac{2^l-1}{2^l} \cdot \frac{2^l-2}{2^l} \cdot \ldots \cdot \frac{2^l-n}{2^l} =$$
$$(1-\frac{1}{2^l})(1-\frac{2}{2^l}) \ldots (1-\frac{n}{2^l}) \geq$$
$$(1-\frac{n}{2^l})^n = (1-\frac{1}{n^{\omega(1)}})^n = 1-\mu(n)$$

where μ is a negligible function in n. In the sequel we therefore focus only on distinct sets of challenges $r_i \in \{0,1\}^l, (r_i^{(1)}, .., r_i^{(n)}) \in \{0,1\}^{nl}$. We show that extraction fails only for a fraction $\frac{1}{n+1}$ of such challenges.

Towards this goals, we define an equivalence class over distinct sets of challenges:

- $\left(a, (a^{(1)}, .., a^{(n)})\right)$ and $\left(b, (b^{(1)}, .., b^{(n)})\right)$ are said to be in the same class if $a = b$, and $(a^{(1)}, .., a^{(n)})$ is a permutation of $(b^{(1)}, .., b^{(n)})$.

Note that for all challenges $\left(a, (a^{(1)}, .., a^{(n)})\right)$ and $\left(b, (b^{(1)}, .., b^{(n)})\right)$ which are in the same class and any prefix view v it holds that if extraction fails (succeeds resp.) when $r_i = a, r_i^{(j)} = a^{(j)}$ for $j \in [n]$, and $view^i_{P'} = v$, then extraction also fails (succeeds resp.) when $r_i = b, r_i^{(j)} = b^{(j)}$ for $j \in [n]$, and $view^i_{P'} = v$. We conclude that either all challenges $(a, \bar{a})$ that belong to a class will result in a successful extraction, or all of them will result in a failed extraction. Furthermore, note that the number of elements in every class is the same.

Now, note that for every class[10] $\left(a, (a^{(1)}, .., a^{(n)})\right)$ and every prefix view v such that extraction fails when $r_i = a, r_i^{(j)} = a^{(j)}$ for $j \in [n]$, and $view^i_{P'} = v$, there exists n other distinct classes for which extraction will succeed, namely $\left(a^{(1)}, (a, a^{(2)}, .., a^{(n)})\right)$, $\left(a^{(2)}, (a^{(1)}, a, .., a^{(n)})\right)$,..., $\left(a^{(n)}, (a^{(1)}, .., a^{(n-1)}, a)\right)$. (Note that all the above classes indeed are distinct since we only consider

[10] We slightly abuse of notation and denote the class by an element that belongs to it.

challenges $a, (a^{(1)}, .., a^{(n)}$ that are all distinct.) Thus, every class has n "companion" classes such that at most one of them will result in a failed extraction. Furthermore, it holds that any two distinct classes that are not each others companions have *disjoint* companion classes; this follows from the fact if two classes $\Big(a, (a^{(1)}, .., a^{(n)}\Big)$ and $\Big(b, (b^{(1)}, .., b^{(n)})\Big)$ have the same companion $\Big(c, (c^{(1)}, .., c^{(n)})\Big)$, then the unordered set $\{b, b^{(1)}, .., b^{(n)}\}$ must be equal to the unordered set $\{a, a^{(1)}, .., a^{(n)}\}$, which means that $\Big(a, (a^{(1)}, .., a^{(n)}\Big)$ and $\Big(b, (b^{(1)}, .., b^{(n)})\Big)$ are each others companions.

We conclude that for any prefix view v, the fraction of distinct challenges $a, (a^{(1)}, .., a^{(n)})$ for which extraction fails if $r_i = a$, $r_i^{(j)} = a^{(j)}$ for $j \in [n]$ and $view^i_{P'} = v$ is

$$\frac{1}{n+1}$$

This in turn means that the fraction of all challenges for which extraction fails is bounded by

$$1 - \Big(1 - \mu(n)\Big)\Big(1 - \frac{1}{n+1}\Big) =$$
$$\frac{1}{n+1} - \frac{\mu(n)}{n+1} + \mu(n) \leq \frac{1}{n+1} + \mu(n)$$

As in the proof of lemma 3, we conclude that the probability that extraction fails in i'th execution

$$\frac{1}{n+1} + \mu(n)$$

Again, as in the proof of lemma 3, this implies that the overall failure probability is bounded by

$$\Big(\frac{1}{n+1} + \mu(n)\Big)^m \leq \Big(\frac{2}{(n+1)}\Big)^{\omega(1)}$$

which is negligible in n.

Witness Indistinguishability. As in the proof of lemma 3 it directly follows that (P', V') is $\mathcal{WI}$ if (P, V) is so. ■

3.2 Computational Knowledge Precision Lemmas

We show how to obtain precise computational proofs of knowledge by sequential repetition of a so-called *computationally special-sound proof*. (Looking ahead, we mention that the protocol of Blum when instantiated with public-coin statistically-hiding commitments is known to be computationally special-sound.)

We extend the definition of special-soundness to a computational notion of soundness as follows. Let (P, V) be a k-round public-coin interactive proof for the language $L \in \mathcal{NP}$ with witness relation R_L, such that the $k-1$-round is a verifier round. Then, (P, V) is said to be *computationally special-sound* if there exists a polynomial-time extractor machine X such that for every polynomial-time machine P^*, and every polynomial $p(\cdot)$, there exists a negligible function μ such that the following holds for every $x \in L$ and every auxiliary input z for P^*. Let $\vec{T} = (T_1, T_2, \ldots T^{p(|x|)}$ denote transcripts in $p(|x|)$ random executions between $P^*(x, z)$ and $V(x)$ where V uses the same randomness for the first $k-2$ rounds (thus, the first $k-2$ rounds in T_i and T_j are the same for all i, j). Then, the probability (over the randomness needed to generate $\vec{T}$ that there exists some

i, j such that the $k-1$'st rounds in T_i and T_j are different, V accepts in both T_i and T_j (i.e., $\text{ACCEPT}_V(x, T_i) = \text{ACCEPT}_V(x, T_j) = 1$), but $X(T_i, T_j)$ does not output a witness $w \in R_L(x)$, is smaller than $\mu(|x|)$. If, furthermore, the predicate ACCEPT_V can be computed in linear time and X has a linear running time, we say that (P, V) is *computationally special-sound with linear extraction.*

We show the following lemmas:

Lemma 5 (Computational Knowledge Precision Lemma - Linear Precision) *Let (P, V) be a computationally special-sound proof system with linear extraction for the language L, with witness relation R_L. Let $m(n) = \omega(\log n)$, and let $(\tilde{P}, \tilde{V})$ denote the m-sequential repetition of (P, V). Then $(\tilde{P}, \tilde{V})$ is a computationally-sound proof of knowledge with linear precision, for the language L with witness relation R_L. If, furthermore (P, V) is (statistical/perfect) $\mathcal{WI}$, then $(\tilde{P}, \tilde{V})$ is so as well.*

Lemma 6 (Computational Knowledge Precision Lemma - Polynomial Precision) *Let (P, V) be a computational special-sound proof system for the language L, with witness relation R_L. Let $m(n) = \omega(1)$, and let $(\tilde{P}, \tilde{V})$ denote the m-sequential repetition of (P, V). If in the execution of (P, V) on common input x the length of the verifier challenge is $\omega(\log |x|)$, then $(\tilde{P}, \tilde{V})$ is a computationally-sound proof of knowledge with polynomial precision, for the language L with witness relation R_L. If, furthermore (P, V) is (statistical/perfect) $\mathcal{WI}$, then $(\tilde{P}, \tilde{V})$ is so as well.*

Proof of Lemma 5 and Lemma 6: Both lemmas follow essentially directly from the proofs of Lemma 4 and Lemma 3. The only point that needs to be addresses it that the special-soundness extractor only is require to function properly when it receives transcripts that have been generated by a polynomial-time machine. Furthermore, this extractor might also fails (with some small probability).

However, since the definition of computationally-sound precise proofs of knowledge only consider computationally-bounded malicious provers P' it follows that also the extractors constructed in Lemma 4 and Lemma 3 are polynomial-time. We conclude that the probability that the special-soundness extractor fails to output a valid witness on input two consistent and accepting transcripts that have been generated by the extractors of Lemma 4 and Lemma 3 is negligible. By the union-bound we thus get that the total failure probability also is negligible. This concludes the proof of Lemma 5 and Lemma 6. ■

3.3 Constructions of WI Precise Proofs of Knowledge

We provide constructions of $\mathcal{WI}$ precise proof of of knowledge for all languages in $\mathcal{NP}$ by combining our knowledge precision lemmas with known $\mathcal{WI}$ proof of knowledge protocols.

3.3.1 $\mathcal{WI}$ Precise Proofs of Knowledge

By combining Lemma 4 and Lemma 3 with the Blum's proof system for Hamiltonicity [11] (relying on [46] and [59]), we obtain:

Theorem 1 *Assume the existence of one-way functions. Then, there exists an (efficient-prover) $\omega(1)$-round $\mathcal{WI}$ statistically-sound proof of knowledge for $\mathcal{NP}$ with polynomial precision. There also exists an (efficient-prover) $\omega(\log n)$-round $\mathcal{WI}$ statistically-sound proof of knowledge for $\mathcal{NP}$ with linear precision.*

Proof:

Polynomial Precision Case. Recall that Blum's proof system for Hamiltonicity [11] is a special-sound proof. Since the proof system is $\mathcal{ZK}$, it is also $\mathcal{WI}$ [27]. Furthermore, since $\mathcal{WI}$ is closed under parallel composition [27], the parallelized version of Blum's protocol (i.e., the protocol resulting from running n parallel copies of the protocol) is also a $\mathcal{WI}$ special-sound proof, which additionally has the property that the length of the verifier challenge in a proof of statements $x \in \{0,1\}^n$, is $\Omega(n)$. Furthermore, it trivially holds due to the construction of the protocol that the verifier will always reject if the prover communicates less than n bits. The first part of the theorem is obtained by combining the above proof system with lemma 4.

Linear Precision Case. It can be seen that, if using an appropriate representation of graphs, Blum's Hamiltonicity protocol is special-sound with linear extraction. The second part of the theorem is obtain by combining this proof system with lemma 3. ■

3.3.2 Statistical-$\mathcal{WI}$ Precise Proofs of Knowledge

By instead combining Lemma 5 and 6 with a statistical $\mathcal{ZK}$ variant Blum's Hamiltonicity protocol (obtained by instantiating the commitments in Blum's protocol with statistically hiding commitment), we instead obtain:

Theorem 2 *Assume the existence of $k(n)$-round public-coin statistically-hiding commitments. Then, there exists an (efficient-prover) $\omega(k(n))$-round statistical-$\mathcal{WI}$ computationally-sound proof of knowledge for $\mathcal{NP}$ with polynomial precision. There also exists an (efficient-prover) $\omega(k(n)\log n)$-round statistical-$\mathcal{WI}$ computationally-sound proof of knowledge for $\mathcal{NP}$ with linear precision.*

Proof: We start by observing that Blum's protocol when instantiated with public-coin statistically-hiding commitments is both

1. computationally special-sound, and
2. statistically $\mathcal{WI}$

The rest of the proof is concluded in the same way as the proof of Theorem 1. ■

3.3.3 Emulatable Precise Proofs of Knowledge

Since all the above-constructed $\mathcal{WI}$ precise proofs of knowledge protocols are public-coin, it directly follows that they are $\mathcal{ZK}$ for the prover with precision $p(n,t) = O(t)$. The following theorems then follow from Theorem 1 and 2 by applying Lemma 2.

Theorem 3 *Assume the existence of one-way functions. Then, there exists an (efficient-prover) $\omega(1)$-round $\mathcal{WI}$ statistically-sound emulatable proof of knowledge for $\mathcal{NP}$ with polynomial precision. There also exists an (efficient-prover) $\omega(\log n)$-round $\mathcal{WI}$ statistically-sound emulatable proof of knowledge for $\mathcal{NP}$ with linear precision.*

Theorem 4 *Assume the existence of $k(n)$-round public-coin statistically-hiding commitments. Then, there exists an (efficient-prover) $\omega(k(n))$-round statistical-$\mathcal{WI}$ computationally-sound emulatable proof of knowledge for $\mathcal{NP}$ with polynomial precision. There also exists an (efficient-prover) $\omega(k(n)\log n)$-round statistical-$\mathcal{WI}$ computationally-sound emulatable proof of knowledge for $\mathcal{NP}$ with linear precision.*

4 Constructions of Precise $\mathcal{ZK}$

In this section we provide our constructions of Precise $\mathcal{ZK}$ protocols. We construct the following Precise $\mathcal{ZK}$ protocols:

1. Statistically Precise $\mathcal{ZK}$ Arguments for $\mathcal{NP}$ (assuming statistically hiding commitments).
2. Computationally Precise $\mathcal{ZK}$ Arguments for $\mathcal{NP}$ (assuming one-way function).
3. Computationally Precise $\mathcal{ZK}$ Proofs for $\mathcal{NP}$ (assuming statistically hiding commitments).
4. Computationally Precise $\mathcal{ZK}$ Proofs for $\mathcal{IP}$ (assuming statistically hiding commitments).
5. Unconditional Statistically Precise $\mathcal{ZK}$ Proofs for specific languages such as Graph Non-Isomorphism.

4.1 Precise $\mathcal{ZK}$ Arguments for $\mathcal{NP}$

We start by showing the following theorem :

Theorem 5 *Assume the existence of $k(n)$-round public-coin statistically-hiding commitments. Then, there exists an (efficient-prover) $k(n)+\omega(1)$-round statistically precise $\mathcal{ZK}$ argument with polynomial precision for every language in $\mathcal{NP}$. There also exists an (efficient-prover) $k(n) + \omega(\log n)$-round statistically precise $\mathcal{ZK}$ argument with linear precision for every language in $\mathcal{NP}$.*

Proof: We begin by constructing a $\mathcal{ZK}$ argument with polynomial precision. This protocol is then modified to obtain a $\mathcal{ZK}$ argument with linear precision.

$\mathcal{ZK}$ arguments with polynomial precision. Recall the protocol of Feige-Shamir. Their protocol proceeds in the following two stages, on common input a statement $x \in \{0,1\}^n$:

1. In Stage 1, the Verifier picks two random strings $r_1, r_2 \in \{0,1\}^n$, and sends their image $c_1 = f(r_1), c_2 = f(r_2)$ through a one-way function f to the Prover. The Verifier furthermore provides a $\mathcal{WI}$ proof of knowledge of the fact that c_1 and c_2 have been constructed properly (i.e., that they are in the image set of f).
2. In Stage 2, the Prover provides a *statistical-$\mathcal{WI}$* proof of knowledge of the fact that *either* x is in the language, *or* (at least) one of c_1 and c_2 are in the image set of f.

We obtain a statistical $\mathcal{ZK}$ argument *PolyPreciseStatZKArg* with precision $p(n,t)$, where p is a polynomial in both n and t, by simply replacing the $\mathcal{WI}$ proofs of knowledge used in Stage 1 of the protocol, with a $\mathcal{WI}$ emulatable proof of knowledge with polynomial precision. If, furthermore, the resulting protocol has the property that V always rejects before Stage 2 is reached unless the prover has communicated less than $|x|$ bits (this for instance directly holds if w.l.o.g. choosing a length preserving one-way function), the protocol has polynomial precision.

More precisely, let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function and let the witness relation $R_{L'}$, where $((x_1, x_2), (y_1, y_2)) \in R_{L'}$ if $f(x_1) = y_1$ or $f(x_2) = y_2$, characterize the language L'. Let the language $L \in \mathcal{NP}$. Protocol *StatPolyPreciseZKArg* for proving that $x \in L$ is described in Figure 1. Note that the protocol relies on the existence of one-way functions, public-coin statistically-hiding commitments and a $\mathcal{WI}$ emulatable proof of knowledge with polynomial precision. However,

since the existence of statistically-hiding commitments, implies the existence of one-way functions, and since by Theorem 3, one-way functions imply the existence of an $\omega(1)$-round $\mathcal{WI}$ emulatable proof of knowledge with polynomial precision, we only require the existence of statistically-hiding commitments. We conclude that the resulting protocol has round complexity $k(n) + \omega(1)$.

Protocol *StatPolyPreciseZKArg*

Common Input: an instance x of a language L with witness relation R_L.

Auxiliary Input for Prover: a witness w, such that $(x, w) \in R_L(x)$.

Stage 1:

V uniformly chooses $r_1, r_2 \in \{0,1\}^n$.

V $\rightarrow$ P: $c_1 = f(r_1), c_2 = f(r_2)$.

V $\rightarrow$ P: a $\mathcal{WI}$ statistically-sound emulatable proof of knowledge with polynomial precision of the statement

either there exists a value r_1 s.t $c_1 = f(r_1)$

or there exists a value r_2 s.t $c_2 = f(r_2)$

The proof of knowledge is with respect to the witness relation R'_L.

Stage 2:

P $\leftrightarrow$ V: a statistical-$\mathcal{WI}$ argument of knowledge of the statement

either there exists values r'_1, r'_2 s.t either $c_1 = f(r'_1)$ or $c_2 = f(r'_2)$.

or $x \in L$

The argument of knowledge is with respect to the witness relation $R_{L \vee L'}(c_1, c_2, x) = \{(r'_1, r'_2, w) | (r'_1, r'_2) \in R_{L'}(c_1, c_2) \vee w \in R_L(x)\}$.

Figure 1: Statistical $\mathcal{ZK}$ argument for $\mathcal{NP}$ with polynomial precision

Proposition 3 *Protocol* StatPolyPreciseZKArg *is a statistical $\mathcal{ZK}$ argument with polynomial precision.*

Proof: Soundness and Completeness of the protocol follows directly from the proof of Feige and Shamir [28] as protocol *StatPolyPreciseZKArg* is a particular instantiation of their protocol. Let us turn to the precise $\mathcal{ZK}$ property. The simulator S for V' proceeds as follows:

1. The simulator S internally incorporates V' and follows the honest prover strategy during the initial commit sent by V'. If V' send an invalid message during the commitment, S halts outputting the view generated for V'

2. Let E denote the precise emulator-extractor for residual verifier V' after V' has committed to $\bar{r}$ (recall that V' acts as a prover in Stage 1 of the protocol).

3. S runs the emulator-extractor E, obtaining a triplet $(view_1, view_2, w' = (r'_1, r'_2))$, where $view_1$ denotes the view of V' (since V' is acting as a prover).

4. If $view_2$ contains a rejecting view, or if $w' \notin R'_L(c_1, c_2)$, S outputs $view_1$ and halts.

5. Otherwise, S performs the following steps:

   (a) S invokes an "internal" copy of V'.

   (b) S feeds the view $view_1$ to V'.

   (c) S then interacts with V' following the honest prover strategy in Stage 2 of the protocol, using $w' = (r'_1, r'_2)$ as witness.

   (d) S finally outputs the view of V' and halts.

Running-time of S. Let v denote the view output by S. We show that the running time of S is polynomial in the running time of V' on the view v. First note that since E is an emulator-extractor with polynomial precision for V', it follows that the time invested by S to generate stage 1 of the view v is polynomial in the running time of V' when feed stage 1 of v. Secondly, since stage 2 of the view v is generated by S emulating the honest prover strategy (using witness w') in an interaction with V', it follows that time invested by S in order to generate stage 2 of v is the time needed to emulate V' in stage 2 of the view v plus the time needed to generate the honest prover messages, which is a polynomial in in $|x|$. Finally, since S only proceeds to generate stage 2 of the view if stage 1 has been successfully completed, it holds that V' must have communicated at least $|x|$ bits (in order to send the strings c_1, c_2), which concludes that the total time invested by S to generate both stage 1 and stage 2 of v is polynomial in the running time of V' on the view v.

Indistinguishability of the simulation. We show that for the following ensembles are statistically close over L

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Towards this goal, consider the following "intermediate" simulator S' that receives a witness y to the statement x. S', on input x, y (and auxiliary input z), proceeds just like S in order to generate Stage 1 of the view, but proceeds as the honest prover in order to generate Stage 2 of the view. Indistinguishability of the simulation by S follows from the following two claims:

Claim 1 *The following ensembles are statistically close over L*

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S'_\bullet(x,(y,z))\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Proof: Assume that E always outputs a witness $w' \in R'_L(x)$ if the view output is accepting. Under this (unjustified) assumption if follows from the perfect emulation condition on E that the view of V' in a real interaction is identical to the output of S'. However, by the precise statistically-sound proof of knowledge property of Stage 1 it follows that the probability that E fails in outputting a witness is negligible. We conclude that the ensembles in the statement of Claim 1 are statistically close. ■

Claim 2 *The following ensembles are statistically close over L*

- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S'_\bullet(x,(y,z))\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Proof: The claim follows directly from the statistical-$\mathcal{WI}$ property[11] of Stage 2 of the protocol, and from the fact that the only difference between S and S' is the choice of the witness used in Stage 2 of the protocol. For completeness, we provide a proof.

Assume for contradiction that the claim is false, i.e., that there exists a deterministic verifier V' (we assume w.l.o.g that V' is deterministic, as it can always receive it random-tape as auxiliary input), a polynomial $g(n)$, and a distinguisher D such that for infinitely many $x \in L$ there exists $y \in R_L(x), z \in \{0,1\}^*$ such that

$$\left|\Pr\left[v \leftarrow S_\bullet(x,z) : D(x,z,v)=1\right] - \Pr\left[v \leftarrow S'_\bullet(x,(y,z)) : D(x,z,v)=1\right]\right| \geq \frac{1}{g(|x|)}$$

Fix generic x, y, z for which this happens. Since S' proceeds exactly as S in Stage 1 of the protocol, there must thus exists a partial view v^1 for V', of only Stage 1 of the protocol, such that D also distinguishes the output of S and S' conditioned on the event that S and S' feed V' the view v^1 as part of its Stage 1 view.

Note that the partial view v^1 defines an instance $x' \in L \vee L'$ that V' expects to hear a proof of, and that the only difference between the executions of S and S' given the view v^1 is the choice of witness used in the proof. We have thus reached a contradiction to the $\mathcal{WI}$ property of Stage 2 of the protocol. ■

■

$\mathcal{ZK}$ arguments with linear precision. We proceed to construct an argument system that is $\mathcal{ZK}$ with linear precision. We obtain the new argument system, called *StatLinPreciseZKArg*, by modifying the previously constructed one, *StatPolyPreciseZKArg*, in the following ways:

1. In Stage 1 of the protocol, V start by sending the string $1^{W(|x|)}$, where $W(|x|)$ denotes the number of computation steps required by P to complete stage 2 of the protocol on input x, given any witness w'. (The prover directly aborts the proof if V sends a string that is shorter.) This message serves as a "zero-knowledge proof" that the malicious verifier has performed roughly as much computation as the honest verifier. (Since we require that simulation is linear in the running time of the malicious verifier, it is imperative that we can simulate also verifiers that run much faster than then honest verifier. This additional message makes it possible to simulate also such verifiers.)

2. In Stage 1 of the protocol, P and V engage in a $\mathcal{WI}$ statistically-sound proof of knowledge with *linear* precision (instead of one with polynomial precision).

[11] We here rely on the fact that the statistical-$\mathcal{WI}$ property holds also for *unbounded* verifiers.

More precisely, let f be a one-way function and let $R_{L'}$, L' be defined as above. Let the language $L \in \mathcal{NP}$. Protocol *StatLinPreciseZKArg* for proving that $x \in L$ is depicted in Figure 2. Since by Theorem 3 the existence of one-way functions (which are implied by the existence of statistically hiding commitments) implies the existence of an $\omega(\log n)$-round $\mathcal{WI}$ emulatable proof of knowledge with linear precision, the resulting protocol has round complexity $k(n) + \omega(\log n)$.

Protocol *StatLinPreciseZKArg*

Common Input: an instance x of a language L with witness relation R_L.

Auxiliary Input for Prover: a witness w, such that $(x, w) \in R_L(x)$.

Stage 1:

Let $W(|x|)$ denotes (an upper bound) on the number of computational steps required by P to complete stage 2 of the protocol on input x, given any witness w'.

$V \rightarrow P$: $1^{W(|x|)}$

P verifies that V sent a string of length $1^{W(|x|)}$. (If not, it aborts.)

V uniformly chooses $r_1, r_2 \in \{0,1\}^n$.

V $\rightarrow$ P: $c_1 = f(r_1), c_2 = f(r_2)$.

V $\rightarrow$ P: a $\mathcal{WI}$ statistically-sound proof of knowledge with linear precision of the statement

either there exists a value r_1 s.t $c_1 = f(r_1)$

or there exists a value r_2 s.t $c_2 = f(r_2)$

The proof of knowledge is with respect to the witness relation R'_L

Stage 2:

P $\leftrightarrow$ V: a statistical-$\mathcal{WI}$ argument of knowledge of the statement

either there exists values r'_1, r'_2 s.t either $c_1 = f(r'_1)$ or $c_2 = f(r'_2)$.

or $x \in L$

The argument of knowledge is with respect to the witness relation $R_{L \vee L'}(c_1, c_2, x) = \{(r'_1, r'_2, w) | (r'_1, r'_2) \in R_{L'}(c_1, c_2) \vee w \in R_L(x)\}$.

Figure 2: Statistical $\mathcal{ZK}$ argument for $\mathcal{NP}$ with linear precision

Proposition 4 *Protocol* StatLinPreciseStatZKArg *is a statistical $\mathcal{ZK}$ argument with polynomial precision.*

Proof: Soundness and Completeness of the protocol follows directly as in the proof of Claim 3. To argue the $\mathcal{ZK}$ with linear precision property consider the same simulator S as in the proof of Claim 3. It directly follows (using the proof of Claim 3) that the output of S is "correctly" distributed. It only remains to analyze the running time of S.

Running-time of S. As in the proof of Claim 3, let v denote the view output by S. Since E is an emulator-extractor with linear precision for V', it follows that the time invested by S to generate stage 1 of the view v is linear in the running time of V' when feed stage 1 of v. Furthermore, note that if V' "completed" stage 1 of the protocol then V' must have spent at least $W(|x|)$ computation steps.

As in the proof of Claim 3) since stage 2 of the view v is generated by S emulating the honest prover strategy (using witness w' extracted in Stage 1) in an interaction with V', it follows that the time invested by S in order to generate stage 2 of v, is the time needed to emulate V' in stage 2 of the view v plus the time needed to generate the honest prover messages. By our assumption that a Turing machine can be emulated at only linear overhead, it follows that the first term is a linear function of the running time of V' in stage 2 of v. The second quantity is (by definition) $W(|x|)$, which is smaller than the total running time of V' on the view v. We conclude that the total time invested by S to generate both stage 1 and stage 2 of v is linear in the running time of V' on the view v. ■

This concludes the proof of Theorem 5. ■

Remark: We note that if we use a precise POK in Stage 2 of protocols *StatPolyPreciseZKArg* and *StatLinPreciseStatZKArg*, we would get a protocol that is both precise $\mathcal{ZK}$ and a precise POK.

4.1.1 Computationally Precise $\mathcal{ZK}$ Arguments from Any One-way Function

Just as in the protocol of Feige and Shamir [28], it follows that if replacing the statistical-$\mathcal{WI}$ proof of knowledge in stage 2 of the protocols *StatPolyPreciseZKArg*, *StatLinPreciseZKArg* with a computational-$\mathcal{WI}$ proof of knowledge, we instead obtain a computational precise $\mathcal{ZK}$ argument. Since constant-round computational-$\mathcal{WI}$ proof of knowledge can be based on the sole assumption of the existence of one-way functions [27, 34, 59, 46], we thus obtain:

Theorem 6 *Assume the existence of one-way functions. Then, there exists an (efficient-prover) $\omega(1)$-round precise computational $\mathcal{ZK}$ argument with polynomial precision, for every language in $\mathcal{NP}$. There also exists an (efficient-prover) $\omega(\log n)$-round precise computational $\mathcal{ZK}$ argument with linear precision, for every language in $\mathcal{NP}$.*

Proof: The proof essentially follows from the proof of Theorem 5. If relying on the same simulator S as in the proof of Theorem 5, the only point that needs to be addressed is the proof of the indistinguishability of the simulation. In particular, since Stage 2 of the protocol is only computational $\mathcal{WI}$ (instead of statistical $\mathcal{WI}$) Claim 2 no longer holds; however, it follows using exactly the same argument that the following claim holds instead, which is sufficient to conclude that the simulation is computationally indistinguishable from the view of the verifier in a true interaction with a prover.

Claim 3 *The following ensembles are computationally indistinguishable over L*

- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S'_\bullet(x,(y,z))\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

■

4.2 Precise $\mathcal{ZK}$ Proofs for $\mathcal{NP}$

We show:

Theorem 7 *Assume the existence of $k(n)$-round statistically hiding commitments. Then, there exists an (efficient-prover) $\omega(k(n))$-round precise computational $\mathcal{ZK}$ proof with polynomial precision, for every language in $\mathcal{NP}$. There also exists an (efficient-prover) $\omega(k(n)\log n)$-round precise computational $\mathcal{ZK}$ proof with linear precision, for every language in $\mathcal{NP}$.*

Proof: We start by constructing a $\mathcal{ZK}$ proof with polynomial precision.

$\mathcal{ZK}$ proofs with polynomial precision. Recall the $\mathcal{ZK}$ proof system for (Graph3Col) of Goldreich and Kahan [33]. Their protocol proceeds in two stages. In the first stage the verifier commits, using a *statistically hiding* commitment, to n pairs of edges in the graph. In the second stage, the prover and the verifier execute n parallel (and using independent random coins) instances of GMW's (Graph3Col) protocol, with the exception that the verifier does not pick random challenges, but instead reveals the edges it committed to in the first stage.

We modify their protocol as follows: In the first stage of the protocol, we additionally let the verifier provide a statistical-$\mathcal{WI}$ computationally-sound emulatable precise proof of knowledge of the values it has committed. (This modification can be seen as a generalization of the protocol of [65, 67].) To obtain a precision $p(n,t)$ that is polynomial in only t, we furthermore require that the verifier must communicate at least $|x|$ bits in order to successfully complete stage 1.

More precisely, let the language $L \in \mathcal{NP}$. Our protocol for proving that $x \in L$ is called *CompPolyPreciseZKProof* and is depicted in Figure 3.

Note that the protocol relies on the existence of statistically-hiding commitment, statistically-binding commitment and a statistically-$\mathcal{WI}$ computationally-sound emulatable proof of knowledge with polynomial precision. However, since the existence of statistically-hiding commitments, implies the existence of one-way functions, which in turns implies the existence of (constant-round) statistically-binding commitments, and since by Theorem 4, the existence of a $k(n)$-round public-coin statistically-hiding commitment implies the existence of a $\omega(k(n))$ round statistical-$\mathcal{WI}$ computationally-sound emulatable proof of knowledge with polynomial precision, we only require the existence of statistically-hiding commitments. We conclude that the resulting protocol has round complexity $\omega(k(n))$.

Proposition 5 *Protocol* CompPolyPreciseZKProof *is a computational $\mathcal{ZK}$ proof with polynomial precision.*

Proof: Note that *CompPolyPreciseZKProof* is a particular instantiation of the protocol of Goldreich and Kahan [33]. This follows since, by the statistical $\mathcal{WI}$ property of the proof in Stage 1 of *PolyPreciseZKProof*, it holds that the whole of Stage 1 is a statistically-hiding commitment.[12] Thus, Soundness and Completeness of *CompPolyPreciseZKProof* follows directly from the proof of Goldreich and Kahan [33]. Let us turn to the precise $\mathcal{ZK}$ property. For a given malicious verifier V', let E denote the precise extractor for V' (recall that V' acts as a prover in Stage 1 of the protocol). The simulator S for V' proceeds as follows:

[12] If not, we could break the hiding property of the original commitment Com, by in exponential-time finding a decommitment to Com, and next emulating the Stage 1 proof using this decommitment as witness. By the Statistical-WIproperty, this emulated view is statistically close to a valid Stage 1 commitment to the same value as Com was a commitment to.

Protocol *CompPolyPreciseZKProof*

Common Input: an instance x of a language L with witness relation R_L.

Auxiliary Input for Prover: a witness w, such that $(x, w) \in R_L(x)$.

Stage 1:

V uniformly chooses $\overline{r} = r_1, r_2, ..., r_n \in \{0,1\}^n$, $s \in \{0,1\}^{poly(n)}$.

V $\rightarrow$ P: $c =$ COM$(\overline{r}; s)$, where COM is a statistically hiding commitment, which has the property that the committer must communicate at least m bits in order to commit to m strings.

V $\rightarrow$ P: a statistical-$\mathcal{WI}$ computationally-sound proof of knowledge with polynomial precision of the statement

there exists values $\overline{r}', s'$ s.t $c =$ COM$(\overline{r}'; s')$

The proof of knowledge is with respect to the witness relation $R'_L(c) = \{(v, s) | c =$ COM$(v; s)\}$.

Stage 2:

P $\leftrightarrow$ V: P and V engage in n parallel executions of the GMW's (3-round) Graph 3-Coloring protocol, where V uses the strings $r_1, .., r_n$ as its challenges:

1. P $\rightarrow$ V: n (random) first messages of the GMW proof system for the statement x.
2. V $\leftarrow$ P: V decommits to $\overline{r} = r_1, .., r_n$.
3. P $\rightarrow$ V: For $i = 1..n$, P computes the answer (i.e., the 3'rd message of the GMW proof system) to the challenge r_i and sends all the answers to V.

Figure 3: Computational $\mathcal{ZK}$ Proof for $\mathcal{NP}$ with Polynomial Precision

1. S runs the emulator-extractor E, obtaining a triplet $(view_1, view_2, w' = (\overline{r}', s'))$, where $view_1$ denotes the view of V' (since V' is acting as a prover).
2. If $view_2$ contains a rejecting view, or if $w' \notin R'_L(x)$, S outputs $view_1$ and halts.
3. Otherwise, S perform the following steps:
   (a) Just as in [33], S generates a "random-looking" execution $(m_1, \overline{r}', m_2)$ of the "parallelized" GMW protocol, where the verifier query $\overline{r}' = \overline{r}$. (This property of the GMW protocol is sometimes called special honest-verifier $\mathcal{ZK}$.)[13]
   (b) S feeds the view $view_1$ to V'.
   (c) S feeds m_1 to V'.
   (d) If V' decommits to $\overline{r}$, S feeds m_2 to V', outputs the view of V' and halts.
   (e) If V' fails to decommit, S outputs the view of V' and halts.
   (f) If V' succeeds in decommit to a different value than $\overline{r}$, S output `fail` and halts.

Running-time of S. Let v denote the view output by S. We show that the running time of S is polynomial in the running time of V' in the view v. First note that since E is an extractor with polynomial precision for V', it follows that the time invested by S to generate stage 1 of the view v is polynomial in the running time of V' when feed stage 1 of v. Secondly, since stage 2 of the view v, is generated by S by emulating the honest prover strategy (using the knowledge of the verifier query $\overline{r}$) in an interaction with V', it follows that time invested by S in order to generate stage 2 of v, is the time needed to emulate V' in stage 2 of the view v plus the time needed to generate the honest prover messages, which is a polynomial in in $|x|$. Finally, since S only proceeds to generate stage 2 of the view if stage 1 has been successfully completed, it holds that V' must have communicated at least $|x|$ bits, which concludes that the total time invested by S to generate both stage 1 and stage 2 of v is polynomial in the running time of V' on the view v.

Indistinguishability of the simulation. We show that the following ensembles are computationally indistinguishable over L.

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Towards this goal, consider the following "intermediate" simulator S' that receives a witness y to the statement x. S', on input x, y (and auxiliary input z), proceeds just like S in order to generate Stage 1 of the view, but proceeds as the honest prover in order to generate Stage 2 of the view. The indistinguishability of the simulation by S follows from the following two claims:

Claim 4 *The following ensembles are statistically close over L*

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

[13] Note that this is possible since it is easy to commit to a coloring such that the two vertices on a *particular* (predetermined) edge have different colors.

- $\left\{S'_\bullet(x,(y,z))\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Proof: The proof is essentially identical to the proof of claim 1; the only difference is that the proof of knowledge protocol in stage 1 is now only computationally-sound.

Assume that E always output a witness $w' \in R'_L(x)$ if the view output is accepting. Under this (unjustified) assumption if follows from the perfect emulation condition on E that view of V' in a real interaction is identical to the output of S'. However, by the precise computationally-sound proof of knowledge property of Stage 1 it follows that the probability that E fails in outputting a witness is negligible. (Note that we here rely on the fact that V' is a polynomial-time machine). We conclude that the ensembles in the statement of Claim 4 are statistically close. ■

Claim 5 *The following ensembles are computationally indistinguishable over L*

- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S'_\bullet(x,(y,z))\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Proof Sketch: We start by noting that it follows directly from the computational-binding property of the commitment scheme used in Stage 1, that the probability of either S or S' outputting `fail` is negligible. It also follows from the special honest-verifier $\mathcal{ZK}$ property of the GMW protocol that the output of $S_\bullet(x,z)$, conditioned of not being `fail`, and the output of $S'_\bullet(x,(y,z))$, conditioned of not being `fail` are computationally indistinguishable. We conclude that $S_\bullet(x,z)$ and $S'_\bullet(x,(y,z))$ are computationally indistinguishable. ■

■

$\mathcal{ZK}$ proofs with linear precision. As for the case of $\mathcal{ZK}$ arguments (see Proposition 4), we obtain a $\mathcal{ZK}$ proof for $\mathcal{NP}$ with linear precision, called Protocol *CompLinPreciseZKProof* by modifying Stage 1 of protocol *CompPolyPreciseZKProof* in the following two ways:

1. V start by sending the string $1^{W(|x|)}$, where $W(|x|)$ denotes the number of computation steps required by S to perform a simulation of Stage 2 of the protocol. (The prover directly aborts the proof if V sends a string that is shorter.)

2. P and V then engage in a statistical-$\mathcal{WI}$ computationally-sound proof of knowledge with *linear* precision (instead of one with polynomial precision).

Since by Theorem 4 the existence of $k(n)$-round statistically hiding commitments implies the existence of an $\omega(k(n)\log n)$-round statistically-$\mathcal{WI}$ computationally-sound emulatable proof of knowledge with linear precision, the resulting protocol has round complexity $\omega(k(n)\log n)$.

It follows exactly as in the proof of Proposition 4 that Protocol *CompLinPreciseZKProof* is computational $\mathcal{ZK}$ with linear precision. This concludes the proof of Theorem 7. ■

4.3 Everything Provable is Provable in Precise $\mathcal{ZK}$

We extend the results from the previous section to show that every language that has an interactive proof also has a $\mathcal{ZK}$ proof with linear precision.

Theorem 8 *Assume the existence of statistically hiding commitments. Then, every language in $\mathcal{IP}$ has a computational $\mathcal{ZK}$ proof with linear precision.*

Proof Sketch: Recall that [10] show that every language having an interactive proof also has a computational $\mathcal{ZK}$ proof. In fact, they provide a transformation from an interactive proofs for a language L, to a $\mathcal{ZK}$ proof for the same language by relying on any $\mathcal{ZK}$ proof for $\mathcal{NP}$ (such as the GMW protocol). We note that if instead relying on a precise $\mathcal{ZK}$ protocol for $\mathcal{NP}$ in their transformation, the resulting protocol will be $\mathcal{ZK}$ with precision $p(n,t)$ where p is a polynomial in both n and t. If furthermore V on common input x is required to communicate at least $|x|$ bits as part of its first message, the protocol is $\mathcal{ZK}$ with polynomial precision.

As in Proposition 4, we obtain a $\mathcal{ZK}$ proof with linear precision, by relying on a $\mathcal{ZK}$ proof with linear precision in the transformation of [10], and by additionally letting the verifier V start by sending the string $1^{W(|x|)}$, where $W(|x|)$ denotes the number of computation steps required by S to perform a simulation of the first stage of the protocol (i.e., the "encrypted" interactive proof).[14] (The prover directly aborts the proof if V sends a string that is shorter.) ■

4.4 Existence of Statistically Precise $\mathcal{ZK}$ Proofs

We provide *unconditional* constructions of precise *statistical $\mathcal{ZK}$ proofs* for certain specific languages. We here exemplify our approach by showing a precise $\mathcal{ZK}$ proof for Graph Non-Isomorphism. Roughly speaking, our construction proceeds in the following steps:

1. We first recast (a variant, due to Benaloh [3], of) Goldreich, Micali and Wigderson's protocol [34] for Graph Non-Isomorphism as an instance of the Feige-Shamir protocol.

2. We then essentially rely on the same construction paradigm as in our previous (conditional) constructions; namely, we use our knowledge precision lemmas to transform a special-sound proof of knowledge into a precise proof of knowledge, and then use the precise proof of knowledge protocol as a sub-protocol to obtain a precise $\mathcal{ZK}$ proof.

4.4.1 Unconditional $\mathcal{WI}$ Precise Proof of Knowledge for a Specific Language

We provide an example of a *statistical-$\mathcal{WI}$ statistically-sound precise proof of knowledge* for a specific language. As mentioned above, this protocol will then be used in order to construct a precise $\mathcal{ZK}$ proof for GraphNonIso.

Consider the language 1of2GraphIso of triplets of graphs G_0, G_1, H, such that H isomorphic to either G_0 or G_1, and the corresponding witness relation $R_{\text{1OF2GRAPHISO}}$ which describes the two isomorphism. We show how to construct a 3-round special-sound $\mathcal{WI}$ proof for $R_{\text{1OF2GRAPHISO}}$ with soundness $\frac{1}{2}$. The protocol (which is a variant of a protocol implicit in [34] and the protocol of Benaloh [3]) is depicted in Figure 4.

Proposition 6 *Protocol* 1of2GraphIsoProof *is a special-sound statistical-$\mathcal{WI}$ proof for* 1of2GraphIso *with witness relation* $R_{\text{1OF2GRAPHISO}}$.

[14] Note that although the prover strategy is not necessarily efficient, the simulator is. Thus, $W(|x|)$ is always a polynomial.

Protocol *1of2GraphIsoProof*

Common Input: an instance G_0, G_1, H of the language 1OF2GRAPHISO.

Auxiliary Input for Prover: a witness w, such that $((G_O, G_1, H), w) \in R_{\text{1OF2GRAPHISO}}(x)$.

P uniformly selects a bit i, and lets C_i be a random isomorphic copy of G_0 and C_{1-i} be a random isomorphic copy of G_1.

P $\rightarrow$ V: C_0, C_1.

V $\rightarrow$ P: a random bit b.

P $\rightarrow$ V:

1. If $b = 0$, P sends the permutation from C_i, C_{1-i} to G_0, G_1.
2. If $b = 1$, P sends the permutation from H to one of C_i, C_{1-i}.

V checks the validity of the permutations received.

Figure 4: Statistically Precise $\mathcal{ZK}$ proof for 1OF2GRAPHISO

Proof: Soundness and Completeness follow directly using the same proof as in [34]. Statistical-$\mathcal{WI}$ follows from the fact that protocol *1of2GraphIsoProof* is honest-verifier perfect zero-knowledge (see [34]), or can be directly argued. ■

By using parallel repetition and an appropriate representation of the graphs, we thus obtain:

Proposition 7 *There exists a 3-round statistical-$\mathcal{WI}$ special-sound proof system (P, V) with linear extraction for* 1OF2GRAPHISO *with witness relation* $R_{\text{1OF2GRAPHISO}}$. *Furthermore, the verifier query in (P, V) for a statement $x \in \{0,1\}^n$ is of length $\Omega(n)$.*

By combining Proposition 7 with Lemma 4 and Lemma 3 we obtain:

Theorem 9 *There exists an $\omega(1)$-round statistical-$\mathcal{WI}$ statistically-sound proof of knowledge for* 1OF2GRAPHISO *with polynomial precision. There also exists an $\omega(\log n)$-round statistical-$\mathcal{WI}$ statistically-sound proof of knowledge for* 1OF2GRAPHISO *with linear precision.*

Since the above-constructed $\mathcal{WI}$ precise proofs of knowledge protocols are public-coin, it directly follows that they are $\mathcal{ZK}$ for the prover with precision $p(n,t) = O(t)$. The following theorem then follows from Theorem 9 by applying Lemma 2.

Theorem 10 *There exists an $\omega(1)$-round statistical-$\mathcal{WI}$ statistically-sound emulatable proof of knowledge for* 1OF2GRAPHISO *with polynomial precision. There also exists an $\omega(\log n)$-round statistical-$\mathcal{WI}$ statistically-sound emulatable proof of knowledge for* 1OF2GRAPHISO *with linear precision.*

4.4.2 Statistically Precise $\mathcal{ZK}$ Proof for Graph Non-Iso

Let GRAPHNONISO denote the language of non-isomorphic graphs.

Theorem 11 *There exists an $\omega(1)$-round statistical $\mathcal{ZK}$ proof for* GraphNonIso *with polynomial precision. There also exists an $\omega(\log n)$-round statistical $\mathcal{ZK}$ proof of knowledge for* GraphNonIso *with linear precision.*

Proof: Let 1of2GraphIso and $R_{\text{1of2GraphIso}}$ be defined as in Section 4.4.1. Consider the protocol depicted in Figure 5 for proving that $x \in$ GraphNonIso. (Note that this protocol does not necessarily have an *efficient* prover strategy. This is potentially unavoidable, as GraphNonIso might not be in $\mathcal{NP}$.)

Protocol *PreciseZKGraphNonIso*

Common Input: an instance G_0, G_1 of the language GraphNonIso.

Stage 1:

V uniformly chooses a bit i and let H be a random isomorphic copy of G_i.

V $\rightarrow$ P: H.

V $\leftrightarrow$ P: V provides a statistically-$\mathcal{WI}$ statistically-sound emulatable proof of knowledge, with polynomial precision, that $G_0, G_1, H \in$ 1of2GraphIso. The proof of knowledge is with respect to the witness relation $R_{\text{1of2GraphIso}}$.

Stage 2:

P $\rightarrow$ V: The bit i' such that H is isomorphic to $G_{i'}$.

V accepts if $i' = i$.

Figure 5: Statistically Precise $\mathcal{ZK}$ proof for GraphNonIso

The following claim concludes the first part of the theorem.

Proposition 8 *Protocol* PreciseZKGraphNonIso *is a statistical $\mathcal{ZK}$ proof for* GraphNonIso *with polynomial precision.*

Proof: Soundness and Completeness of the protocol follows as in [34]. $\mathcal{ZK}$ with polynomial precision follows directly from the statistically-sound emulatable proof of knowledge with polynomial precision property of Stage 1. ■

In order to obtain a $\mathcal{ZK}$ proof with linear precision we proceed in exactly the same way as in the proof of Theorem 5. ■

4.4.3 Other Unconditional Statistically Precise $\mathcal{ZK}$ Proofs

The same approach as above can directly be applied to Goldwasser, Micali and Rackoff's [38] protocol for Quadratic Non-Residuosity, QNR. Furthermore, by instead relying on a protocol of Micciancio, Ong, Sahai, and Vadhan [56] (extending [65] and [57]) we can obtain statistical precise $\mathcal{ZK}$ proof with polynomial and linear precision for all problems in $\mathbf{SD}^1_{1/2}$ [57, 69]. (The general, i.e., non-restricted, Statistical Difference problem is complete for statistical $\mathcal{ZK}$ [69]). We note that although GraphNonIso and QNR actually reduces to $\mathbf{SD}^1_{1/2}$, the protocol resulting from relying on the protocol of [56] require a "large" round-complexity and non-efficient provers, whereas our direct approach avoids this.

5 Black-Box Lower Bounds for Precise $\mathcal{ZK}$

We show that only $\mathcal{ZK}$ proof/argument systems for 'trivial" languages can have black-box simulators with precision $p(n,t)$, where p is a polynomial in both n and t. Intuitively, this lower bound follows from the following observations:

- A black-box simulator S for a zero-knowledge proof of a non-trivial language *must* rewind the verifier at least once (otherwise the simulator could be used as a cheating prover).
- Since the simulator only uses the verifier as a black-box it is *oblivious* of the running time of the verifier on the view output. Furthermore, it is oblivious of the running time of the verifier in the rewound execution. Therefore, if the malicious verifier decides how long to run based on (in a randomized way) the queries that the simulator sends, we can with relatively high probability end up in a situation where the simulator outputs a view in which the verifier runs very fast, but the running time of the verifier in the rewound execution is long.

We proceed to a formal treatment relying on the above intuition.

5.1 Definition of Black-Box Precise $\mathcal{ZK}$

Our definition of black-box precise $\mathcal{ZK}$ is a straight-forward restriction of the definition of precise $\mathcal{ZK}$ to only allow for black-box simulators, in analogy with the definition of black-box $\mathcal{ZK}$ (see Definition 14 in Section B.4). For simplicity (and since we are proving a lower), we only state the definition for the weakest form of precise $\mathcal{ZK}$, namely computational precise $\mathcal{ZK}$. In fact, to make the lower-bound even stronger we present a definition where we only require that with *overwhelming probability*, the running-time of the simulator is related to that of the verifier.

Definition 6 (Weak Precise Black-box $\mathcal{ZK}$) *Let (P,V) be an interactive proof (argument) system for the language $L \in \mathcal{NP}$ with the witness relation R_L, and let and $p : N \times N \to N$ be a monotonically increasing function. We say that (P,V) is* weak computational black-box $\mathcal{ZK}$ with precision $p(n,t)$ *if there exists a probabilistic oracle machine S such that for every probabilistic polynomial-time interactive machine V', the following two conditions hold:*

1. *The following two ensembles are computationally indistinguishable over L.*
   - $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)]\right\}_{x\in L, y\in R_L(x), z,r\in\{0,1\}^*}$
   - $\left\{S_\bullet^{V'_r(x,z)}(x)\right\}_{x\in L, y\in R_L(x), z,r\in\{0,1\}^*}$
2. *There exists a negligible function, such that for every $x \in L$ and every $z, r \in \{0,1\}^*$ it holds that*
$$\Pr\left[\text{STEPS}_{S_\bullet^{V'_r(x,z)}(x)} \leq p(|x|, \text{STEPS}_{V'}(S_\bullet^{V'_r(x,z)}(x))\right] \geq 1 - \mu(n)$$
   (We emphasize that in the above expression the two occurrences of $S_\bullet$ refer to the same *random variable.)*

5.2 The Lower Bound

We prove the following theorem:

Theorem 12 *Let (P, V) be an m-round weak black-box computational $\mathcal{ZK}$ interactive proof (or argument) with precision $p(n,t) \in poly(n,t)$ for the language L. Then,*

$$L \in \boldsymbol{BPTIME}[O(p(n, \text{TIME}_V(n)))]$$

Before proceeding to the proof, we make the following remarks:

1. First, note that Theorem 12 shows that only languages in $\mathcal{BPP}$ have black-box zero-knowledge proofs with polynomial precision. However, recall that precise zero-knowledge proofs with *linear precision* might be interesting also for languages in $\mathcal{BPP}$ or $\mathcal{P}$. Concerning such proof systems, Theorem 12 states that the *honest* verifier of the zero-knowledge proof (argument) system needs to perform "essentially" as much computation as is needed to decide the language, completely trivializing the proof system.

2. Also, note that Theorem 12 is "tight" in the sense that if we relax the restriction on *polynomial* precision, then black-box simulation *can* be useful. In particular, Pass' interactive arguments for $\mathcal{NP}$ [63] (which are based on the existence of one-way functions with sub-exponential hardness) are black-box zero-knowledge with *quasi-polynomial* precision.[15]

Proof of Theorem 12: Suppose, for contradiction, that there exists an m-round (where $m = m(n)$) interactive argument (P, V), that has a black-box simulator S with precision $p(n,t)$. Consider the malicious verifier V' defined below. V' proceeds just as V, except for the following differences:

1. V' receives as auxiliary input the description of an $(m+1)$-wise independent hash function $H : \{0,1\}^{l(n)} \to \{0,1\}^n$, where $l(n)$ denotes an upper-bound on the length of a transcript of the protocol (P, V).

2. Let $T(n)$ be a polynomial in n, soon to be determined. Each time V' is given a query, V' applies H to the partial transcript up until this query (including the query) to generate a random number s. Based on this randomness, V' decides to, with probability $1/m$ "pause" for $\Big(p(n, T(n))\Big)^2$ steps before proceeding as V, and otherwise directly proceeds as V. More precisely, if the first $\log m$ positions of s are zero, then V' runs $\Big(p(n, T(n))\Big)^2$ dummy instructions and then proceeds to do what V would. Otherwise it directly does what V would do.

   The polynomial $T(n)$ used above is defined as the upperbound on the time invested by V', *except* for the "pauses" (i.e., $T(n)$ is essentially TIME_V plus the time needed to evaluate the hasfunction H, m times). Note that in case V' does not "pause" in a view, its running time is thus trivially bounded by $T(n)$.

We show that unless $L \in \mathbf{BPTIME}[O(p(n, \text{TIME}_V(n)))]$ there exist some non-negligible function $g(n)$ such that with probability at least $g(n)$, $S^{V'}$ outputs a view in which V' runs at most $T(n)$ steps but which took S at least $\Big(p(n, T(n))\Big)^2$ steps to generate. Towards this goal we start by showing the following claim.

[15] Those protocols are $\mathcal{ZK}$ with precision $p(n,t) = O(n^{\text{polylog}_n} + t)$. As an additional interesting feature they are constant-round, whereas we don't know if constant-round zero-knowledge protocols with polynomial precision can be constructed.

Claim 6 *Unless $L \in \boldsymbol{BPTIME}[O(p(n, \text{TIME}_V(n)))]$, there exists some non-negligible function $g(n)$ such that with probability at least $g(n)$, $S^{V'}$ queries V' on a message m' that is not part of the view output by S.*

Proof: Assume, for contradiction, that S with overwhelming probability (i.e., with probability $1 - \mu(n)$, where μ is a negligible function), *only* queries V' on messages that are part of the view output by S. We show how this implies that S combined with V can be used to decide the language. More precisely, consider the deciding machine D defined as follows. On input an instance x, D performs the following steps:

1. D picks a random tape r for V, and executes $view \leftarrow S_\bullet^{V_r(x,z)}(x)$. If S attempts to perform more than $p(n, \text{TIME}_V(n))$ computational steps, halt outputting $\perp$.
2. Unless D has already halted, it finally outputs $\text{OUT}_V(view)$ (i.e., it outputs 1 if and only if V accepts in the view $view$.)

We start by noting that the running-time of D on input an instance $x \in \{0,1\}^n$ is $O(p(n, \text{TIME}_V(n)))$. We proceed to show that D decides L. First note that it directly follows from the validity of S that D outputs $\perp$ only with negligible probability. (Recall that D only outputs $\perp$ when S attempts to take more than $O(p(n, \text{TIME}_V(n)))$ steps. Since the running-time of V is upper-bounded by $\text{TIME}_V(n)$ this thus only happens with negligible probability). In the sequel of the analysis we therefore disregard this rare event.

- Given an instance $x \in L$, it follows directly from the $\mathcal{ZK}$ and completeness properties of (P, V) that, except with negligible probability, $S_\bullet^{V_r(x,z)}(x)$ outputs a view in which V accepts; we conclude that $D_\bullet(x) \to 1$ except with negligible probability.
- Given an instance $x \notin L$ it instead holds that except with negligible probability, S outputs a view in which V rejects. If this was not the case S could be used as a cheating prover. This follows since S only queries the verifier on a message that is not part of the view output, with negligibly small probability.

  More precisely, assume for contradiction that $S_\bullet^{V_r(x,z)}(x)$ outputs a view in which V accepts with non-negligible probability. As in [33], we may without loss of generality make two simplifying assumption about S: 1) it never asks the same query twice to its oracle, and 2) whenever it queries the oracle with a transcript, it has previously queried it on all partial transcripts.

  We now construct a cheating prover P' as follows.

  1. P' internally runs $S_\bullet(x)$.
  2. Whenever S makes a query to its oracle, externally forwards the last prover messages in the query and provides S with the answer received back.

  We start by noting that as long as S only asks queries that are consistent with a single execution of (P, V), the view of S in the emulation by P' is *identical* to the view of S when interacting with V'. (Note that we here rely on the two simplifying assumptions about S so ensure that the queries are forwarded out in the "right" order.) Since, except with negligible probability, S in an execution with V' only asks questions that are consistent with a single execution of (P, V), it holds that also in the emulation by P', S, except with negligible probability, only asks questions that are consistent with a single execution of (P, V). We conclude that the

view of S in the emulation by P' is *statistically close* to the view of S when interacting with V', which implies that the success probability of P' is non-negligible. This contradicts the Soundness of (P, V).

■

Relying on Claim 6 we show the following claim, which concludes the proof of Theorem 12.

Claim 7 *There exists some auxiliary input z for V' such that the probability that S takes more than $\Big(p(n, T(n))\Big)^2$ computation steps in order to generate a view v in which the running time of $V'(z)$ is $T(n)$, is*

$$O\Big(g(n)\frac{1}{m}(1-\frac{1}{m})^m\Big) \approx O\Big(\frac{g(n)}{m \cdot e}\Big)$$

Proof: We show that for a *random* choice of an $(m+1)$-wise independent hash function H, it holds that the probability that S invests at least $\Big(p(n, T(n))\Big)^2$ computation steps S in order to output a view in which the running time of $V'(H)$ is $T(n)$, is

$$O\Big(g(n)\frac{1}{m}(1-\frac{1}{m})^m\Big)$$

By an averaging argument, this concludes that there exists at least one auxiliary input $z = H$ satisfying the conditions of the claim.

Note that due to the construction of V' it holds that for any *fixed* view of V, the probability (over the choice of the hash function H) that the running time of V' is at most $T(n)$ steps is

$$(1-\frac{1}{m})^m$$

This follows since V' uses the $(m+1)$-wise independent hash function to decide whether to "pause" or not, and from the fact that V' only applies this function m times.

By Claim 6 it holds that with probability $g(n)$, S feeds a query to V' that is not part of the view v output. Since this query (by definition) is different to the queries in the view, it holds that with (independent) probability

$$\frac{1}{m}$$

V' will run in time $p(n, T(n))^2$ when feed this query. (Here, independence follows since V' uses an $(m+1)$-independent hash function, and since we are only applying this function on $m+1$ different points). We conclude that with probability

$$O\Big(g(n)\frac{1}{m}(1-\frac{1}{m})^m\Big)$$

S takes time $\Big(p(n, T(n))\Big)^2$ in order to generate a view in which V' takes at most $T(n)$ steps. ■

■

Extensions to precise proofs of knowledge. We note that (assuming the existence of one-way function) our black-box lower bound also extends to rule out the existence of $\mathcal{WI}$ precise proofs of knowledge for $\mathcal{NP}$. This follows from the fact that (assuming the existence of one-way functions) we show how to construct precise $\mathcal{ZK}$ arguments for $\mathcal{NP}$ given a $\mathcal{WI}$ precise proof of knowledge for $\mathcal{NP}$.

6 Precise Encryption

We provide a definition of precise public-key encryption. For generality, we consider security under CPA, CCA1 or CCA2 attacks.

Definition 7 (Encryption Scheme) *A triple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme, if* Gen *and* Enc *are p.p.t. algorithms and* Dec *is a deterministic polynomial-time algorithm,*

1. Gen *on input* 1^n *produces a tuple* $(\mathrm{PK}, \mathrm{SK})$*, where* PK, SK *are the public and private keys,*
2. $\mathsf{Enc} : \mathrm{PK} \times \{0,1\}^* \to \{0,1\}^*$ *runs on input a public key* PK *and a message* $m \in \{0,1\}^*$ *and produces a ciphertext* c*,*
3. $\mathsf{Dec} : \mathrm{SK} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ *runs on input* (SK, c) *and produces either a message* $m \in \{0,1\}^*$ *or a special symbol* $\perp$*,*
4. *There exists a polynomial* $p(k)$ *and a negligible function* $\mu(k)$ *such that for* every message m*, and* every random tape r_e*,*

$$\Pr[r_g \xleftarrow{R} \{0,1\}^{p(k)};\ (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n; r_g);\ \mathsf{Dec}_{\mathrm{SK}}(\mathsf{Enc}_{\mathrm{PK}}(m; r_e)) \neq m] \leq \mu(k).$$

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme, A,S PPTs, and M a non-uniform PPT. Then, let $\mathsf{real}_{\pi,ATK,M,A}(1^n, z)$, $\mathsf{ideal}_{\pi,S}(1^n, z)$ denote the probability distributions resulting from the followings experiments.

$\mathsf{real}_{\pi,ATK,M,A}(z)$:	$\mathsf{ideal}_{\pi,S}(1^n, z)$:
$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}_\bullet(1^n)$	$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}_\bullet(1^n)$
$(m, hist) \leftarrow M_\bullet^{O_1}(\mathrm{PK})$	$(m, hist) \leftarrow M_\bullet^{O_1}(\mathrm{PK})$
$c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; \bullet)$	$view \leftarrow S_\bullet(1^n, \mathrm{PK}, hist, z)$
$x \leftarrow A_\bullet^{O_2}(1^n, \mathrm{PK}, c, hist, z)$	$x \leftarrow A_\bullet(view)$
Output x.	Output x.

If ATK=CPA, then $O_1 = O_2 = \epsilon$. If ATK=CCA1 then $O_1 = \mathsf{Dec}_{\mathrm{SK}}(\cdot)$, $O_2(c) = \epsilon$. If ATK=CCA2 then $O_1 = \mathsf{Dec}_{\mathrm{SK}}(\cdot)$, $O_2(c) = \mathsf{Dec}_{\mathrm{SK}}(c)$ if $c \neq y$ and otherwise $\perp$.

Definition 8 (Precise Encryption) *We say that the encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* perfectly/statistically/computationally secure *with* precision p *under* ATK*, if, for every PPT ITM* A*, there exists a probabilistic algorithm* S *such that for every non-uniform PPT* M*, the following holds.*

1. *The following two ensembles are identical/statistically close/computationally indistinguishable over* $n \in N$*.*
   (a) $\left\{\mathsf{real}_{\pi,ATK,M,A}(1^n, z)\right\}_{n \in N, z \in \{0,1\}^*}$
   (b) $\left\{\mathsf{ideal}_{\pi,S}(1^n, z)\right\}_{n \in N, z \in \{0,1\}^*}$
2. *For every* $n \in N$*, every* $z \in \{0,1\}^*$*,*

$$\Pr\Big[(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}_\bullet(1^n); (m, hist) \leftarrow M_\bullet^{O_1}(\mathrm{PK}); r \leftarrow \{0,1\}^\infty \ : \\ \mathrm{STEPS}_{S_r(1^n, \mathrm{PK}, hist, z)} \leq p(n, \mathrm{STEPS}_A(S_r(1^n, \mathrm{PK}, hist, z)))\Big] = 1$$

The notion of precise private-key encryption is obtained by adapting experiments real and ideal in the straight-forward way (namely, removing PK, letting $c \leftarrow \mathsf{Enc}_{\mathrm{SK}}(m)$, and additionally providing M, A with an encryption oracle, in the case of CCA security).

Our main result is stated below.

Theorem 13 *If π is a CPA-secure public-key encryption scheme that on input a message of length n outputs a* pseudo-random *ciphertext of length $l(n)$, and furthermore the length $l(n)$ can be computed in time $O(n)$, then π is (computationally) secure with precision $p(n,t) = O(t)$ under CPA.*

Proof: We construct a simulator $S(\mathrm{PK}, z, hist)$ for A. Assume, for simplicity, that A has $1^n, \mathrm{PK}, z, hist$ hard-coded in its description; they can be easily handled as S can simply send the appropriate bit of any of them to A whenever A request to read it; this only incurs a linear overhead.

$S(1^n, \mathrm{PK}, z, hist)$ next feeds random bits to $A(1^n, \mathrm{PK}, z, hist)$ until A halts, or until it has feed A n bits. If A has halted, simply output the view of A; otherwise compute $l(n)$ (note that $l(n) > n$ or else we could not encrypt n-bit messages) and continue feeding A random bits until it halts, or until $l(n)$ bits has been fed. Finally, output the view of A.

As we assumed that a TM can emulate another at only linear overhead, we conclude that the running-time of S is linear in the running-time of A (given any view). Additionally, it follows from the pseudorandom ciphertext property of π that the simulation by S is valid. ■

Using the same proof we also get:

Theorem 14 *The one-time pad is perfectly secure encryption scheme with precision $p(n,t) = O(t)$ under CPA.*

We point out that not necessarily every perfectly secure encryption scheme has linear precision. Consider for instance a family of functions $\{f_s\}$, such that $f_s : \{0,1\}^{|x|} \to \{0,1\}^*$ is 1-1 for every s; assume further that given s one can compute in polynomial-time (say, in time$|s|^5$) both f and f^{-1}. Such a family could very well give rise to a perfectly secure encryption scheme (when picking a random s as a key), yet it is not clear that one can simulate an encryption in time smaller than $|s|^5$, even if the adversary runs much faster.

CCA2-secure Encryption We show how to turn any CCA2-secure encryption scheme into one that is CCA2-secure with linear precision; the transformation relies on a padding argument and should only be taken as a feasibility results (showing that linear precision can be obtained). We leave open the question of constructing "practical" CCA2-secure precise encryption schemes.

Theorem 15 *Assume the existence of CCA2-secure encryption schemes. Then there exist a CCA2-secure encryption scheme with precision $p(n) = O(n)$.*

Proof: Let $\pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA2 secure encryption scheme. Let t_{Gen}, $t_{\mathsf{Enc}}(n)$, $t_{\mathsf{Dec}}(n)$ denote the respective running-times of $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$. Let $g(n)$ be a function that is:

1. lower-bounded by $t_{\mathsf{Gen}}(n) + t_{\mathsf{Enc}}(n) + t_{\mathsf{Dec}}(n)$;
2. upper-bounded by a polynomial;
3. computable in time linear in n.

Note that since $t_{\mathsf{Gen}}(n) + t_{\mathsf{Enc}}(n) + t_{\mathsf{Dec}}(n)$ is a polynomial, such a functions is easy to find.

Consider the encryption scheme $\pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ defined as follows.

- $\mathsf{Gen}'(\cdot)$: run $\text{PK}, \text{SK} \leftarrow \mathsf{Enc}_{\text{PK}}(m)$, let $\text{PK}' = 1^{g(n)}||\text{PK}$, $\text{SK}' = \text{SK}$, and output (PK', SK').
- $\mathsf{Enc}'_{\text{PK}'}(m)$: interpret PK' as $1^{g(n)}||\text{PK}$, run $c \leftarrow \mathsf{Enc}_{\text{PK}}(m)$ and output $c' = 1^{g(n)}||c$.
- $\mathsf{Dec}'_{\text{SK}}(c')$: check if c' starts with $g(n)$ leading 1's. If so, interpret c' as $1^{g(n)}||c$ and output $\mathsf{Dec}_{\text{SK}}(c)$. Otherwise output $\perp$.

We show that π' is CCA2-secure with precision $p(n) = O(n)$. Again, we assume for simplicity that the adversary A has $1^n, \text{PK}, z, hist$ hard-coded in its description. Consider the simulator S that start by feeding A 1's as part of its public-key and as part of the ciphertext, until A halts, or until it has feed A n bits. If A has halted, simply output the view of A; otherwise compute $g(n)$ (note that $g(n) > n$ or else we could not encrypt n-bit messages) and continue feeding A 1's until it halts, or until $g(n)$ bits has been fed. Again, if at any point A halts, S does so as well (outputting the view of A). Once A has read all the ones, S let $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$, $\text{PK}' = 1^{g(n)}||\text{PK}$, $c' \leftarrow \mathsf{Enc}'_{\text{PK}'}(0^n)$ and feeds c' to A. Next S answers all decryption queries in the following way: it reads the string $\tilde{c'}$ sent out by A bit by bit; if it starts by $1^{g(n)}$ leading ones, and $\tilde{c'} \neq c'$, let $\tilde{m} \leftarrow \mathsf{Dec}'_{\text{SK}}(\tilde{c'})$ and output $\tilde{m}$; otherwise output $\perp$. Finally output the view of A in this execution. It directly follows from the construction that the running-time of S is linear in the running-time of A given any view. Additionally, it direct follows from the (standard) $CCA2$ security of π that the view output by S is correctly distributed. ■

7 Precise Secure Computation

We provide a precise analogue of the notion of secure computation [34]. We consider a malicious *static* computationally bounded (i.e., probabilistic polynomial-time) adversary that is allowed to corrupt a $t(m)$ of the m parties—that is, before the beginning of the interaction the adversary corrupts a subset of the players that may deviate arbitrarily from the protocol. The parties are assumed to be connected through authenticated point-to-point channels. For simplicity, we here assume that the channels are synchronous.

A multi-party protocol problem for m parties $P_1, ..., P_m$ is specified by a random process that maps vectors of inputs to vectors of outputs (one input and one output for each party). We refer to such a process as a n-ary functionality and denote it $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$, where $f = (f_1, ..., f_m)$. That is, for every vector of inputs $\bar{x} = (x_1, ..., x_m)$, the output-vector is a random variable $(f_1(\bar{x}), ..., f_n(\bar{x}))$ ranging over vectors of strings. The output of the i'th party (with input x_i) is defined to be $f_i(\bar{x})$.

As usual, the security of protocol π for computing a function f is defined by the real execution of π with an "ideal" execution where all parties directly talk to a trusted party.

The ideal execution. Let $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$ be a n-ary functionality, where $f = (f_1, ..., f_m)$. Let S be a probabilistic polynomial-time machine (representing the ideal-model adversary) and let $I \subset [m]$ (the set of corrupted parties) be such that for every $i \in I$, the adversary S controls party P_i. The ideal execution of f with security parameter n, inputs $\bar{x} = (x_1, ..., x_m)$ and auxiliary input z to S, denoted $\mathsf{ideal}_{f,I,S}(n, \bar{x}, z)$, is defined as the output vector of the parties $P_1, .., P_m$ and the adversary S, resulting from the process below:

Each party P_i receives it input x_i from the input vector $\bar{x} = (x_1, .., x_m)$. Each honest party P_i sends x_i to the trusted party. Corrupted parties P_i can instead send any arbitrary value $x'_i \in \{0,1\}^{|x_i|}$ to the trusted party. When the trusted third party has received messages x'_i from all parties (both

honest and corrupted) it sets $\bar{x}' = (x'_1, .., x'_m)$, computes $y_i = f(\bar{x}')$ and sends y_i to P_i for every $i \in I$. When the adversary sends the message (send-output, i) to the trusted party, the trusted party delivers y_i to P_i. Finally, each honest party P_i outputs the value y_i received by the trusted party. The corrupted parties, and also S, may output any arbitrary value.

Additionally, let $\text{view}^P_{V'}(x)_{f,I,S}(n, \bar{x}, z)$ denote (probability distribution) describing the view of S in the above experiment.

The real execution. Let f, I be as above and let Π be a multi-party protocol for computing f. Furthermore, let A a probabilistic polynomial-time machine. Then, the real execution of Π with security parameter n, inputs $\bar{x} = (x_1, ..., x_m)$ and auxiliary input z to A, denoted $\mathsf{real}_{\Pi,I,A}(n, \bar{x}, z)$, is defined as the output vector of the parties $P_1, .., P_m$ and the *view* of the adversary A resulting from executing protocol π on inputs $\bar{x}$, when parties $i \in I$ are controlled by the adversary $A(z)$.

Definition 9 (Precise Secure Computation) *Let f and Π be as above, and $p : N \times N \times N \to N$ a monotonically increasing function. Protocol Π is said to t-*securely compute *f with precision p and* perfect/statsitical/computational security *if for every probabilistic polynomial-time real-model adversary A, there exists an probabilistic ideal-model adversary S, such that for every $m \in N$, $\bar{x} = (x_1, ..., x_m) \in (\{0,1\}^*)^m$ and every $I \subset [m]$, with $|I| \leq t$, the following conditions hold:*

1. *The following two ensembles are identical/statistically close/computationally indistinguishable, over $n \in N$.*
   (a) $\left\{\mathsf{ideal}_{f,I,S}(k, \bar{x}, z)\right\}_{k \in N, z \in \{0,1\}^*}$
   (b) $\left\{\mathsf{real}_{\Pi,I,A}(k, \bar{x}, z)\right\}_{k \in N, z \in \{0,1\}^*}$
2. *For every $n \in N$, every $\bar{x} \in (\{0,1\}^*)^m$, and every $z \in \{0,1\}^*$,*

$$\Pr\left[view \leftarrow \text{view}^P_{V'}(x)_{f,I,S}(n, \bar{x}, z) : \text{STEPS}_S(view) \leq p(|x|, \text{STEPS}_A(S(view)\right] = 1.$$

Theorem 16 *Assume the existence of enhanced trapdoor permutations. Then, for any m-party functionality f there exists a protocol Π that $(m-1)$-securely computes f with precision $p(n) = O(n)$ and computational security.*

Proof Sketch: We only provide a proof sketch. Our construction proceeds in two steps:

1. We first show how to construct a precise secure computation protocol when having access to an *idealized* $\mathcal{ZK}$ proof of knowledge functionality [16]. We here rely on the protocols of [34, 17, 64] and on padding. This is possible since given access to a idealized $\mathcal{ZK}$ functionality, we can construct a straight-line simulatable secure computation protocol (see [17, 64]); this means that the overhead $o(t, n)$ of the simulator is some fixed polynomial $p(n)$ only in the security parameter n, independent of the running-time of the verifier t (plus the linear time needed to simulate the adversary). Now, if we pad all protocol messages with $1^{p(n)}$, we can make sure that the adversary spends at least $p(n)$ steps before it will see any "real" protocol messages. Thus, we get linear precision.

2. In the next step we simply implement the idealized $\mathcal{ZK}$ proof of knowledge functionality with a protocol that is both precise $\mathcal{ZK}$ and a precise proof of knowledge.

■

By appropriately modifying the protocols of [8] and [68] (using padding), we instead get

Theorem 17 *For any m-party functionality f there exists a protocol Π that $(m/2-1)$-securely computes f with precision $p(n) = O(n)$ and statistical security (assuming a broadcast channel). For any m-party functionality f there exists a protocol Π that $(m/3-1)$-securely computes f with precision $p(n) = O(n)$ and perfect security.*

These protocol additionally satisfies a stronger definition, where *fairness* is guaranteed; we omit the definition and refer the reader to [32].

References

[1] W. Aiello, J. Håstad. Statistical Zero-Knowledge Languages can be Recognized in Two Rounds. *JCSS*. Vol. 42(3), pages 327–345, 1991

[2] B. Barak. How to go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.

[3] J.D. Benaloh. Cryptographic Capsules: A disjunctive primitive for interactive protocols. In *Crypto86*, Springer LNCS 263, pages 213–222, 1987.

[4] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *JCSS*, Vol. 36, pages 254–276, 1988.

[5] B. Barak and O. Goldreich. Universal Arguments and their Applications. *17th CCC*, pages 194–203, 2002.

[6] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *CRYPTO'92*, Springer (LNCS 740), pages 390–420, 1993.

[7] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. In *34th STOC*, pages 484–493, 2002.

[8] D. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20'th STOC*, pages 1–10, 1988.

[9] M. Bellare, R. Impagliazzo and M. Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th FOCS*, pages 374–383, 1997.

[10] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian ,S. Micali and P. Rogaway. Everything provable is provable in zero-knowledge. In *Crypto88*, Springer LNCS 0403, pages 37-56, 1988.

[11] M. Blum. How to prove a Theorem So No One Else Can Claim It. *Proc. of the International Congress of Mathematicians,* Berkeley, California, USA, pages 1444–1451, 1986.

[12] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *20th STOC*, pages 103–112, 1988

[13] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, Vol. 37, No. 2, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.

[14] M. Blum, A. De Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM J. Computing*, 20(6):1084–1118, 1991.

[15] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *34th STOC*, pages 494–503, 2002.

[16] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Crypto2001*, Springer LNCS 2139, pages 19–40, 2001.

[17] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Computation. In *34th STOC*, pages 494–503,2002.

[18] D. Chaum and H. van Antwerpen. Undeniable Signatures. In *Crypto89*, Springer LNCS 435, pages. 212–216, 1989.

[19] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Crypto94*, Springer LNCS 839, pages. 174–187, 1994.

[20] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EuroCrypt00*, Springer LNCS 1807, pages 418–430, 2000.

[21] Y. Dodis and S. Micali. Parallel Reducibility for Information-Theoretically Secure Computation, In *Crypto00*, Springer LNCS 1880 74–92, 2000.

[22] I. Damgård, T. Pedersen and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. In *Crypto93*, Springer-Verlag LNCS Vol. 773, pages 250–265, 1993.

[23] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *30th STOC*, pages 409–418, 1998.

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

[25] U. Feige, D. Lapidot and A. Shamir. Multiple Noninteractive Zero Knowledge Proofs under General Assumptions. *Siam Jour. on Computing 1999*, Vol. 29(1), pages 1–28.

[26] U. Feige, A. Fiat and A. Shamir. Zero Knowledge Proofs of Identity. *Journal of Cryptology*, Vol. 1, pages 77–94, 1988.

[27] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

[28] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In *Crypto89*, Springer LNCS 435, pages. 526–544, 1989.

[29] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Crypto86*, Springer LNCS 263, pages 181–187, 1987.

[30] M. Fischer, S. Micali, and C. Rackoff. A Secure Protocol for the Oblivious Transfer. *Journal of Cryptology*, 9(3): 191–195, 1996.

[31] O. Goldreich. *Foundations of Cryptography – Basic Tools*. Cambridge University Press, 2001.

[32] O. Goldreich. *Foundations of Cryptography – Basic Applications*. Cambridge University Press, 2004

[33] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Jour. of Cryptology*, Vol. 9, No. 2, pages 167–189, 1996.

[34] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38(1), pp. 691–729, 1991.

[35] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Jour. of Cryptology*, Vol. 7, No. 1, pages 1–32, 1994.

[36] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No 2, pages 270–299, 1984.

[37] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC 85*, pages 291–304, 1985.

[38] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Jour. on Computing*, Vol. 18(1), pp. 186–208, 1989.

[39] S. Goldwasser, S. Micali and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks. *SIAM Jour. on Computing*, Vol. 17, No. 2, pp. 281–308, 1988.

[40] V. Goyal, A. Jain, R. Ostrovsky. Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In *Crypto10*, Springer LNCS 6223, pages 277–294, 2010.

[41] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *Crypto11*, Springer LNCS 6841, pages 297315, 2011.

[42] S. Goldwasser, M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *18'th STOC*, pages 59–68, 1986.

[43] I. Haitner, M. Nguyen, S. Ong, O. Reingold and S. Vadhan. Statistically-Hiding Commitments and Statistical Zero-Knowledge Arguments from Any One-Way Function. To appear in *Siam Journal of Computing*, 2010.

[44] S. Halevi and S. Micali. Conservative Proofs of Knowledge. MIT/LCS/TM-578, May 1998.

[45] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Crypto96*, Springer LNCS 1109, pages 201–215, 1996.

[46] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. *SIAM Jour. on Computing*, Vol. 28 (4), pages 1364–1396, 1999.

[47] J. Halpern and R. Pass. Game Theory with Costly Computation: Formulation and Applications to Protocol Security. In *ICS 2010*, pages 120–142, 2010.

[48] J. Halpern and R. Pass. I Don't Want to Think about it Now: Decision Theory with Costly Computation. In *KR 2010*.

[49] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th STOC*, pages 723–732, 1992.

[50] J. Kilian and E. Petrank. Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds. In *33rd STOC*, pages 560–569, 2001.

[51] J. Katz and Y. Lindell. Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs. In *2nd TCC*, Springer-Verlag (LNCS 3378), pages 128-149, 2005.

[52] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Crypto01*, Springer LNCS 2139, pages 171–189, 2001.

[53] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. In *35th STOC*, pages 683-692, 2003.

[54] S. Micali. CS Proofs. *SIAM Jour. on Computing*, Vol. 30 (4), pages 1253–1298, 2000.

[55] S. Micali and R. Pass. Local Zero Knowledge. In *38th STOC*, 2006.

[56] D. Micciancio, S. Ong, A. Sahai, S. Vadhan. Concurrent Zero Knowledge without Complexity Assumptions. In *3st TCC*, pages 1–20, 2006.

[57] D. Micciancio, S. Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In *Crypto03*. Springer LNCS 2729, pages. 282–298, 2003.

[58] S. Micali and P. Rogaway. Secure computation. Unpublished manuscript, 1992. Preliminary version in *Crypto91*, Springer LNCS 576, pages 392–404, 1991.

[59] M. Naor. Bit Commitment using Pseudorandomness. *Jour. of Cryptology*, Vol. 4, pages 151–158, 1991.

[60] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Perfect Zero-Knowledge Arguments for NP Using any One-Way Permutation. *Jour. of Cryptology*, Vol. 11, pages 87–108, 1998.

[61] Y. Oren. On the Cunning Power of Cheating Verifiers: Some Observations about Zero-Knowledge Proofs. In *28th FOCS*, pages 462–471, 1987.

[62] O. Pandey, R. Pass, A. Sahai, W. Tseng, and M. Venkitasubramaniam. Precise Concurrent Zero Knowledge. In *EuroCrypt08*, Springer LNCS 4965, pages 397–414, 2008.

[63] R. Pass. Simulation in Quasi-polynomial Time and its Application to Protocol Composition. In *EuroCrypt03*, Springer LNCS 2656, pages 160–176, 2003.

[64] R. Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In *36th STOC*, 2004, pages 232-241, 2004.

[65] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent Zero-Knowledge with Logarithmic Round Complexity. In *43rd FOCS*, pages 366–375, 2002.

[66] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer LNCS 1592, pages 415–431, 1999.

[67] A. Rosen. A note on constant-round zero-knowledge proofs for NP. In *1st TCC*, pages 191–2002, 2004.

[68] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multi-party Protocols with Honest Majority. In *21st STOC*, pages 73–85, 1989.

[69] A. Sahai and S. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[70] A. Shamir. IP = PSPACE. In *31st FOCS*, pages 11–15, 1990.

[71] M. Tompa, H. Woll. Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information. In *28th FOCS*, pages 472–482, 1987.

[72] S. Vadhan. *A Study of Statistical Zero-Knowledge Proofs.* PhD thesis, MIT, 1999.

A Basic Notation

A.1 General Notation

We employ the following general notation.

Integer and String representation. We denote by N the set of natural numbers: 0, 1, 2, ...
Unless otherwise specified, a natural number is presented in its binary expansion (with no *leading* 0s) whenever given as an input to an algorithm. If $n \in N$, we denote by 1^n the unary expansion of n (i.e., the concatenation of n 1's). We denote by $\{0,1\}^n$ the set of n-bit long string, by $\{0,1\}^*$ the set of binary strings, and by $[n]$ the set $\{1,..,n\}$.

We denote the concatenation of two strings x and y by $x|y$ (or more simply by xy). If α is a binary string, then $|\alpha|$ denotes α's length and $\alpha_1 \cdots \alpha_i$ denotes α's i-bit prefix.

Probabilistic notation. We employ the following probabilistic notation from [39]. We focus on probability distributions $X : S \to R^+$ over finite sets S.

Probabilistic assignments. If D is a probability distribution and p a predicate, then "$x \stackrel{R}{\leftarrow} D$" denotes the elementary procedure consisting of choosing an element x at random according to D and returning x, and "$x \stackrel{R}{\leftarrow} D \mid p(x)$" denotes the operation of choosing x according to D until $p(x)$ is true and then returning x.

Probabilistic experiments. Let p be a predicate and $D_1, D_2, \ldots$ probability distributions, then the notation $\Pr[x_1 \stackrel{R}{\leftarrow} D_1;\ x_2 \stackrel{R}{\leftarrow} D_2;\ \ldots : p(x_1, x_2, \ldots)]$ denotes the probability that $p(x_1, x_2, \ldots)$ will be true after the ordered execution of the probabilistic assignments $x_1 \stackrel{R}{\leftarrow} D_1;\ x_2 \stackrel{R}{\leftarrow} D_1;\ \ldots$

New probability distributions. If D_1, D_2, ... are probability distributions, the notation $\{x \stackrel{R}{\leftarrow} D_1; y \stackrel{R}{\leftarrow} D_2; \cdots : (x, y, \cdots)\}$ denotes the new probability distribution over $\{(x, y, \cdots)\}$ generated by the ordered execution of the probabilistic assignments $x \stackrel{R}{\leftarrow} D_1$, $y \stackrel{R}{\leftarrow} D_2, \cdots$.

Probability ensembles. Let I be a countable index set. A *probability ensemble indexed by* I is a vector of random variables indexed by I: $X = \{X_i\}_{i \in I}$.

In order to simplify notation, we sometimes abuse of notation and employ the following "shortcut": Given a probability distribution X, we let X denote the random variable obtained by selecting $x \leftarrow X$ and outputting x.

Algorithms. We employ the following notation for algorithms.

Deterministic algorithms. By an algorithm we mean a Turing machine. We only consider *finite* algorithms, i.e., machines that have some fixed upper-bound on their running-time (and thus always halt). If M is a deterministic algorithm, we denote by $\text{STEPS}_{M(x)}$ the number of computational steps taken by M on input x. We say that an algorithm M has time-complexity $\text{TIME}_M(n) = t(n)$, if $\forall x \in \{0,1\}^*$ $\text{STEPS}_{M(x)} \leq t(|x|)$. (Note that time complexity is defined as an upper-bound on the running time of M *independently* of its input.)

Probabilistic algorithms. By a probabilistic algorithms we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, the notation "$M_r(x)$" denotes the output of the M on input x when receiving r as random tape.

We let the notation "$M_\bullet(x)$" denote the probability distribution over the outputs of M on input x where each bit of the random tape r is selected at random and independently(note that this is a well-defined probability distribution since we only consider algorithms with finite running-time.)

Oracle algorithms. Given two algorithms M, A, we let $M^A(x)$ denote the output of the algorithm M on input x, when given oracle access to A.

Emulation of algorithms. In counting computational steps, we assume that an algorithm M, given the code of a second algorithm A and an input x, can emulate the computation of A on input x with only linear overhead.

Negligible functions. The term "negligible" is used for denoting functions that are asymptotically smaller than the inverse of any fixed polynomial. More precisely, a function $\nu(\cdot)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n, it holds that $\nu(n) < n^{-c}$.

A.2 Protocol Notation

We assume familiarity with the basic notions of an *Interactive Turing Machine* [38] (ITM for brevity) and a *protocol*. Briefly, an ITM is a Turing Machine with a read-only *input* tape, a read-only *auxiliary input* tape, a read-only *random* tape, a read/write *work-tape*, a read-only communication tape (for receiving messages) a write-only communication tape (for sending messages) and finally an *output* tape. The content of the input (respectively auxiliary input) tape of an ITM A is called *the input* (respectively *auxiliary input*) *of* A and the content of the output tape of A, upon halting, is called *the output of* A.

A protocol (A, B) is a pair of ITMs that share communication tapes so that the (write-only) send-tape of the first ITM is the (read-only) receive-tape of the second, and vice versa. The computation of such a pair consists of a sequence of rounds $1, 2, ...$ In each round only one ITM is active, and the other is idle. A round ends with the active machine either halting—in which case the protocol ends—or by it entering a special *idle* state. The string m written on the communication tape in a round is called the *message sent* by the active machine to the idle machine.

In this paper we consider protocols (A, B) where both ITMs A, B receive the *same* string as input (but not necessarily as auxiliary input); this input string will be denoted the *common input* of A and B.

We make use of the following notation for protocol executions.

Rounds. In a protocol (A, B), a round $r \in N$ is denoted an A-round (respectively B-round) if A (respectively B) is active in round r in (A, B). We say that a protocol has $r(n)$ rounds (or simply is an $r(n)$-round protocol) if the protocol (A, B) consists of $r(n)$-rounds of communication between A and B when executed on common input $x \in \{0,1\}^n$.

Executions, transcripts and views. Let M_A, M_B be vectors of strings $M_A = \{m_A^1, m_A^2, ...\}$, $M_B = \{m_B^1, m_B^2, ...\}$ and let $x, r_1, r_2, z_1, r_2 \in \{0,1\}^*$. We say that the pair $((x, z_1, r_1, M_A), (x, z_2, r_2, M_B))$ is an execution of the protocol (A, B) if, running ITM A on common input x, auxiliary input z_1 and random tape r_1 with ITM B on x, z_2 and r_2, results in m_A^i being the i'th message *read* by A and in m_B^i being the i'th message *read* by B, and r_1, r_2 are the parts of

the random tapes consumed by A and B respectively. We also denote such an execution by $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$.

In an execution $((x, z_1, r_1, M_A), (x, z_2, r_2, M_B)) = (V_A, V_B)$ of the protocol (A, B), we call V_A the *view of* A (in the execution), and V_B the *view of* B. We let $\text{VIEW}_1[A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)]$ denote A's view in the execution $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$ and $\text{VIEW}_2[A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)]$ B's view in the same execution. (We occasionally find it convenient referring to an execution of a protocol (A, B) as a *joint view* of (A, B).)

In an execution $((x, z_1 r_1, M_A), (x, z_2, r_2, M_B))$, the pair (M_A, M_B) is called the transcript of the execution.

Outputs of executions and views. If e is an execution of a protocol (A_1, A_2) we denote by $\text{OUT}_i(e)$ the output of A_i, where $i \in \{1, 2\}$. Analogously, if v is the view of A, we denote by $\text{OUT}(v)$ the output of A in v.

Random executions. We denote by $A_\bullet(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$, $A_{r_1}(x, z_1) \leftrightarrow B_\bullet(x, z_2)$ and $A_\bullet(x, z_1) \leftrightarrow B_\bullet(x, z_2)$ the probability distribution of the random variable obtained by selecting each bit of r_1 (respectively, each bit of r_2, and each bit of r_1 and r_2) randomly and independently, and then outputting $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$. The corresponding probability distributions for VIEW and OUT are analogously defined.

Counting ITM steps. Let A be an ITM and $v = (x, z, r, (m_1, m_2, ..m_k))$. Then by $\text{STEPS}_A(v)$ we denote the number of computational steps taken by A running on common input x, auxiliary input z, random tape r, and letting the ith message received be m_i.

Time Complexity of ITMs. We say that an ITM A has time-complexity $\text{TIME}_A(n) = t(n)$, if for every ITM B, every common input x, every auxiliary inputs z_a, z_b, it holds that $A(x, z_a)$ *always* halts within $t(|x|)$ steps in an interaction with $B(x, z_b)$, regardless of the content of A and B's random tapes). Note that time complexity is defined as an upperbound on the running time of A *independently* of the content of the messages it receives. In other words, the time complexity of A is the *worst-case* running time of A in *any* interaction.

B Preliminaries

B.1 Indistinguishability

The following definition of (computational) indistinguishability originates in [36].

Definition 10 (Indistinguishability) *Let X and Y be countable sets. Two ensembles $\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be* computationally indistinguishable over X*, if for every probabilistic "distinguishing" algorithm D whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $x \in X, y \in Y$:*

$$| \ Pr[a \leftarrow A_{x,y} : D(x, y, a) = 1] - \ Pr[a \leftarrow B_{x,y} : D(x, y, b) = 1]| < \nu(|x|)$$

$\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be statistically close *over X if the above condition holds for all (possibly unbounded) algorithms D.*

B.2 Interactive Proofs and Arguments

We state the standard definitions of interactive proofs (introduced by Goldwasser, Micali and Rackoff [38]) and arguments (introduced by Brassard, Chaum and Crepeau [13]).

Definition 11 (Interactive Proof (Argument) System) *A pair of interactive machines (P, V) is called an* interactive proof system *for a language L if machine V is polynomial-time and the following two conditions hold with respect to some negligible function $\nu(\cdot)$:*

- Completeness: *For every $x \in L$ there exists a (witness) string y such that*

$$\Pr\left[\text{OUT}_V[P_\bullet(x,y) \leftrightarrow V_\bullet(x)] = 1\right] = 1$$

- Soundness: *For every $x \notin L$, every interactive machine B and every $y \in \{0,1\}^*$*

$$\Pr\left[\text{OUT}_V[P_\bullet(x,y) \leftrightarrow V_\bullet(x)] = 1\right] \leq \nu(|x|)$$

In case that the soundness condition is required to hold only with respect to a computationally bounded prover, the pair (P, V) is called an interactive argument *system.*

Definition 11 can be relaxed to require only soundness error that is bounded away from $1-\nu(|x|)$. This is so, since the soundness error can always be made negligible by sufficiently many parallel repetitions of the protocol. However, in the case of interactive arguments, we do not know whether this condition can be relaxed. In particular, in this case parallel repetitions do not necessarily reduce the soundness error (cf. [9]).

B.3 Commitment Schemes

Commitment schemes are the digital equivalent of physical envelopes. They enable a first party, referred to as the *sender*, to commit itself to a value while keeping it secret from a second party, the *receiver*; this property is called *hiding*. Furthermore, the commitment is *binding*, and thus in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase. The opening phase traditionally consists of the sender simply sending the receiver the value v it committed to, as well as the random coins r it used. The receiver accepts the opening to v if the messages it received during the committing phase are produced by running the honest sender algorithm on input v and the random tape r.

Commitment schemes come in two different flavors, *perfectly-binding* and *perfectly-hiding*.

Perfect-binding. In a perfectly-binding commitments, the binding property holds against unbounded adversaries, while the hiding property only holds against computationally bounded adversaries. Loosely speaking, the perfectly-binding property asserts that the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that commitments to any two different values are computationally indistinguishable; actually, in most applications (and in particular for the construction of zero-knowledge proofs) we require that the indistinguishability of commitments holds even when the distinguisher receives an auxiliary "advice" string (this is sometimes called non-uniform computational hiding).

For simplicity, we present a definition of a commitment scheme for enabling a sender to commit to a *single* bit.

Definition 12 (Perfectly-binding commitment) *A perfectly-binding bit commitment scheme is a pair of probabilistic polynomial-time interactive machines (S, R) satisfying the following properties:*

- Perfect Binding: *For all $r_1, r_2, r' \in \{0,1\}^*, n \in N$ it holds that*

$$\text{VIEW}_2[S_{r_1}(1^n, 0) \leftrightarrow R_{r'}(1^n)] \neq \text{VIEW}_2[S_{r_2}(1^n, 1) \leftrightarrow R_{r'}(1^n)]$$

- Computational Hiding: *For every probabilistic polynomial-time ITM R' the following ensembles are computationally indistinguishable over N*
  - $\left\{\text{VIEW}_2[S_\bullet(1^n, 0) \leftrightarrow R'_\bullet(1^n, z)]\right\}_{n \in N, z \in \{0,1\}^*}$
  - $\left\{\text{VIEW}_2[S_\bullet(1^n, 1) \leftrightarrow R'_\bullet(1^n, z)]\right\}_{n \in N, z \in \{0,1\}^*}$

Above, the variable n is a parameter determining the security of the commitment scheme.

Perfect-hiding. In perfectly-hiding commitments, the hiding property holds against unbounded adversaries, while the binding property only holds against computationally bounded adversaries. Loosely speaking, the perfectly-hiding property asserts that commitments to any two different values are identically distributed. The computational-binding property guarantees that no polynomial time adversary algorithm is able to construct a commitment that can be opened in two different ways; again, for our applications, we actually require that the binding property holds also when providing the adversary with an "advice" string (this property is sometimes called non-uniform computational binding). We omit a formal definition of perfectly-hiding commitments and refer the reader to [31].

Statistical Binding/Hiding. We mention that it is often convenient to relax the perfectly-binding or the perfectly-hiding properties to only statistical binding or hiding. Loosely speaking, the *statistical-binding* property asserts that with overwhelming probability (instead of probability 1) over the coin-tosses of the receiver, the transcript of the interaction fully determines the committed value. The *statistical-hiding* property asserts that commitments to any two different values are statistically close (i.e., have negligible statistical difference, instead of being identically distributed).

Existence of Commitment Schemes. Non-interactive perfectly-binding commitment schemes can be constructed using any 1–1 one-way function (see Section 4.4.1 of [31]). Allowing some minimal interaction (in which the receiver first sends a single message), statistically-binding commitment schemes can be obtained from any one-way function [59, 46]. Perfectly-hiding commitment schemes can be constructed from any one-way permutation [60] and statistically-hiding commitment can be constructed from any one-way function [43]. *Constant-round* schemes are only known to exist under stronger assumptions. Specifically, perfectly hiding commitment can be constructed assuming the existence of a collection of certified clawfree permutations [33] (see also [31], Section 4.8.2.3), and statistically-hiding commitments can be constructed under the potentially weaker assumption of collision-resistant hash functions [22, 45]. All the above commitments are public coin—that is, the receiver only uses *public* random coins.

B.4 Zero Knowledge

We recall the standard definition of $\mathcal{ZK}$ proofs. Loosely speaking, an interactive proof is said to be *zero-knowledge* ($\mathcal{ZK}$) if a verifier V learns nothing beyond the validity of the assertion being proved, it could not have generated on its own. As "feasible" computation in general is defined though the notion of probabilistic polynomial-time, this notion is formalized by requiring that the output of every (possibly malicious) verifier interacting with the honest prover P can be "simulated" by a probabilistic expected polynomial-time machine S (a.k.a. the *simulator*). The idea behind this definition is that whatever V^* might have learned from interacting with P, he could have learned by himself by running the simulator S.

The notion of $\mathcal{ZK}$ was introduced and formalized by Goldwasser, Micali and Rackoff in [37, 38]. We present their definition below.[16]

Definition 13 ($\mathcal{ZK}$) *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P,V) an interactive proof (argument) system for L. We say that (P,V) is* perfect/statistical/computational *$\mathcal{ZK}$, if for every probabilistic polynomial-time interactive machine V' there exists a probabilistic algorithm S whose expected running-time is polynomial in the length of its first input, such that the following ensembles are identical/statistically close/computationally indistinguishable over L.*

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_\bullet(x,z)]\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$
- $\left\{S_\bullet(x,z)\right\}_{x\in L, y\in R_L(x), z\in\{0,1\}^*}$

Black-box Zero-knowledge. One can consider a particularly "well-behaved" type of $\mathcal{ZK}$ called *black-box* $\mathcal{ZK}$. Most known $\mathcal{ZK}$ protocols (with the exception of [2]) and all "practical" $\mathcal{ZK}$ protocols indeed satisfy this stronger notion. Loosely speaking, an interactive proof is black-box $\mathcal{ZK}$ if there exists a (universal) simulator S that uses the verifier V' as a black-box in order to perform the simulation. More precisely (following [31])

Definition 14 (Black-box $\mathcal{ZK}$) *Let (P,V) be an interactive proof (argument) system for the language $L\in\mathcal{NP}$ with the witness relation R_L. We say that (P,V) is* perfect/statistical/computational black-box *$\mathcal{ZK}$, if there exists a probabilistic expected polynomial time oracle machine S such that for every probabilistic polynomial-time interactive machine V', the following two ensembles are identical/statistically close/computationally indistinguishable over L.*

- $\left\{\text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)]\right\}_{x\in L, y\in R_L(x), z,r\in\{0,1\}^*}$
- $\left\{S_\bullet^{V'_r(x,z)}(x)\right\}_{x\in L, y\in R_L(x), z,r\in\{0,1\}^*}$

At first sight the definition of black-box $\mathcal{ZK}$ might seems very restrictive: the simulator is supposed to act as the prover, except that the simulator does not have a witness, and is required to runs in polynomial-time! Note, however that the simulator has an important advantage that the prover does not have—namely that it can *rewind* and restart the verifier. Indeed, this seemingly small advantage is sufficient to perform an efficient simulation, without knowing a witness.

[16] The definition we present here appears in the journal version [38] or [37]. It differs from the original definition of [37] in that "simulation" is required to hold also with respect to all auxiliary "advice"-string $z\in\{0,1\}^*$, where both V^* and S are allowed to obtain z as auxiliary input. The authors of [38] mention that this refinement was independently suggested by the Oren [61], Tompa and Woll [71] and the authors.

Knowledge Tightness. Goldreich, Micali and Wigderson [34], and more recently Goldreich [31] proposes the notion of knowledge tightness as a refinement of $\mathcal{ZK}$. Knowledge tightness is *aimed at measuring the "actual security"* of a $\mathcal{ZK}$ proof system, and is defined as the ratio between the expected running-time of the simulator and the (worst-case) running-time of the verifier [31].[17] More precisely,

Definition 15 *Let $t : N \to N$ be a function. We say that a $\mathcal{ZK}$ proof for L has* knowledge-tightness $t(\cdot)$ *if there exists a polynomial $p(\cdot)$ such that for every probabilistic polynomial-time verifier V' there exists a simulator S (as in Definition 13) such that for all sufficiently long $x \in L$ and every $z \in \{0,1\}^*$ we have*

$$\frac{\mathbf{Exp}\left[\text{STEPS}_{S_\bullet(x,z)}\right] - p(|x|)}{\text{TIME}_{V'(x,z)}} \leq t(|x|)$$

where $\text{TIME}_{V'(x,z)}$ denotes an upper-bound on the running time of V' on common input x and auxiliary input z when receiving arbitrary messages.

Since black-box simulators only query the oracle they have access to an (expected) polynomial number of times, it directly follows that black-box $\mathcal{ZK}$ protocols have polynomial knowledge tightness. Furthermore, many known $\mathcal{ZK}$ protocols have constant knowledge tightness.

We emphasize, however, that the knowledge tightness of $\mathcal{ZK}$ proof systems only refers to the overhead of the simulator with respect to the *worst-case* running time of the verifier.

B.5 Witness Indistinguishability

The notion of Witness Indistinguishability ($\mathcal{WI}$) was introduced by Feige and Shamir in [27] as a weaker alternative to zero-knowledge. Intuitively an interactive proof of an $\mathcal{NP}$ relation, in which the prover uses one of several secret witnesses is $\mathcal{WI}$ if the verifier can not tell what witness the prover has used.

Note that $\mathcal{WI}$ proofs of statements with multiple witnesses provide the guarantee that the (whole) witness used by the prover is not revealed, as this would breach $\mathcal{WI}$. Also note that $\mathcal{WI}$ provides no guarantees when considering proofs of statements with a *single* witness, i.e., such proofs might reveal the whole witness; as such $\mathcal{WI}$ is a significantly weaker property than $\mathcal{ZK}$. Nevertheless, $\mathcal{WI}$ proofs have proved very useful in the design of zero-knowledge protocols, e.g., [28, 25, 66, 2].

We proceed to a formal definition (following [31]),

Definition 16 (Witness Indistinguishability) *Let (P,V) be an interactive proof for the language $L \in \mathcal{NP}$, and R_L be a fixed witness relation for L. We say that (P,V) is $\mathcal{WI}$ for R_L if for every probabilistic polynomial-time algorithm V' and every two sequences $W^1 = \{w_x^1\}_{x\in L}$ and $W^2 = \{w_x^2\}_{x\in L}$, such that $w_x^1, w_x^2 \in R_L(x)$, the following two ensembles are computationally indistinguishable over L.*

[17] To be precise, the authors of [34] define the tightness of zero-knowledge proof as the ratio between the expected running-time of S and the *expected* running-time of V, where the latter expectation is taken only over the random-coins of V, and not over the messages V receives. In other words, in the notation of [34] the expected running-time of V denotes the *worst-case* expected running-time of V in *any* interaction (i.e., an upper-bound on the expected running-time of V that holds when V is receiving all possible messages.) The definition of [31], on the other hand, defines the tightness as the ratio between the expected running-time of S and an upper bound on the running-time of V taken also over all possible random-tapes (as well as all possible messages). Note that this difference is insubstantial as we without loss of generality can consider only *deterministic* malicious verifiers that receive their random-coins as part of their auxiliary input.

- $\left\{\text{VIEW}_2[P_\bullet(x, w_x^1) \leftrightarrow V'_\bullet(x, z)]\right\}_{x\in L, z\in\{0,1\}^*}$
- $\left\{\text{VIEW}_2[P_\bullet(x, w_x^2) \leftrightarrow V'_\bullet(x, z)]\right\}_{x\in L, z\in\{0,1\}^*}$

We further say that (P, V) is statistically (perfectly) $\mathcal{WI}$ for R_L if the above ensembles are statistically close (identically distributed) for every (possibly unbounded) verifier V'.

Remark: Our definitions of statistical and perfect $\mathcal{WI}$ indistinguishability is slightly stronger than the standard ones in that we require indistinguishability for all (possibly unbounded) verifiers, whereas standard definitions only quantify over polynomial-time verifiers. We note however that all known constructions of statistical (perfect) $\mathcal{WI}$ proofs satisfy also our stronger notion.

B.6 Proofs of Knowledge

The notion of a proof of knowledge was intuitively introduced in the paper by Goldwasser, Micali and Rackoff [37] and was formalized by Feige, Fiat and Shamir [26] and Tompa and Woll [71]. The definition was further refined by Bellare and Goldreich [6]. Loosely speaking, an interactive proof is a proof of knowledge if the prover convinces the verifier that it *possesses*, or can *feasibly compute*, a witness for the statement proved. Again, as "feasible" computation is defined through the notion of probabilistic polynomial-time, this notion is formalized by requiring the existence of a *probabilistic polynomial-time* "extractor"-machine that can, given the description of any (malicious) prover that succeeds in convincing the honest verifier, readily compute a valid witness to the statement proved.

We proceed to the actual definition of a proof of knowledge. Our definition follows most closely that of Feige [24] (which in turn follows that of Tompa and Woll [71]).

Definition 17 (Proof of knowledge) *Let (P, V) be an interactive proof system for the language L. We say that (P, V) is a* proof of knowledge *for the witness relation R_L for the language L it there exists a probabilistic expected polynomial-time machine E (called extractor) and a negligible function $\nu(n)$ such that for every probabilistic polynomial-time machine P', every statement $x \in \{0,1\}^n$, every random tape $r \in \{0,1\}^*$ and every auxiliary input $z \in \{0,1\}^*$,*

$$Pr[\text{OUT}_2[P'_r(x, z) \leftrightarrow V_\bullet(x)] = 1] \;\leq\; Pr[E^{P'_r(x,z)}(x) \in R_L(x)] + \nu(n)$$

C Known Non Black-box Simulators are Not Precise

In this section we review why known non black-box simulation techniques, due to Barak [2], result in a non-precise simulation. We start by reviewing Barak's $\mathcal{ZK}$ protocol and then turn to discuss its simulator.

Review of Barak's protocol. The protocol of Barak requires the use of Universal Arguments [5], which are a variant of CS-proofs introduced by Micali [54]. Such proofs systems are used in order to provide "efficient" proofs to statements of the form $y = (M, x, t)$, where y is considered to be a true statement if M is a non-deterministic machine that accepts x within t steps.

Let UARG be a Universal Argument. Let $T : N \to N$ be a "nice" function that satisfies $T(k) = k^{\omega(1)}$. A high-level overview of Barak's protocol is depicted in Figure 6.

Protocol *BarakZK*

Common Input: an instance x of a language L with witness relation R_L.

Auxiliary Input for Prover: a witness w, such that $(x, w) \in R_L$.

Stage 0:

$V \to P$: Send $h \stackrel{R}{\leftarrow} \mathcal{H}_k$.

Stage 1:

$P \to V$: Send $c = \mathrm{Com}(0^k)$.

$V \to P$: Send $r \in \{0,1\}^{|x|}$.

Stage 2: (Proof Body)

$P \leftrightarrow V$: A $\mathcal{WI}$ *UARG* proving the OR of the following two statements:

1. There exists $w \in \{0,1\}^{\mathrm{poly}(|x|)}$ so that $R_L(x, w) = 1$.
2. c is a commitment to a hash using the function h, of the program Π, such that $\Pi(c) = r$ within $T(|x|)$ steps.

Figure 6: Barak's Non Black-Box ZK Argument for $\mathcal{NP}$

As shown in [2], Barak's protocol is computationally sound, under appropriate assumptions on the hash function h (either by assuming that h is collision resistant against circuits of size $\omega(T(n))$, or by assuming that h is constructed by combing a specific tree-hashing approach with any standard collision resistant hash function [5]).

Simulation of Barak's protocol. Given access to the verifier's code (or, equivalently, to the verifier's next message function), the protocol can be simulated without making use of rewinding: To perform simulation, the simulator commits to the verifier's next-message function (instead of committing to zeros). The verifier's next message function is then a program whose output, on input c is r; this provides the simulator with a valid "fake" witness to use in Stage 2 of the protocol.

The Simulation is Not Precise. It is easy to see that the above simulation is not precise: Consider a verifier V' that has a very long auxiliary input tape, but most of the time only accesses a small portion of it. The simulator will always commit to the *whole* description of V' (including the whole auxiliary input tape) and will thus always take long time, while V' might run fast a large portion of the time. (In fact, the running-time of the simulator, will be polynomial in the *worst-case* running-time of the verifier, whereas we require that it is polynomial in the *actual* time of the verifier.)