# A PRECISE COMPUTATIONAL APPROACH TO KNOWLEDGE

by

Rafael Pass

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author ........[illegible]........
Electrical Engineering and Computer Science
June 2006

Certified by ........[illegible]........
Silvio Micali
Professor
Thesis Supervisor

Accepted by ........[illegible]........
Arthur C. Smith
Chairman, Department Committee on Graduate Studies

There's no sense in being precise when you don't even know what you're talking about.

John von Neumann

# Contents

# List of Figures

# Acknowledgments

FIRST AND FOREMOST, I wish to thank Silvio Micali, my advisor and mentor. His daring creativity, colorfulness and energy, combined with knowledge, wisdom and empathy, makes every meeting him a joy. I am deeply grateful to him for his ability to understand me and give me freedom and confidence, while at the same time, in critical moments, providing sincere and determined opinions, helping me making the right decisions. I am also very grateful for his patience and for the long hours he spent teaching me how to express (scientific) thoughts in an aesthetic way.

My interest in Cryptography originated during my studies with Johan Håstad. His self-confidence, precise comments and clear view, helped me organize my perpetually changing chaotic thinking. Johan was always available to listen to my half-baked ideas and has an amazing way of extracting the core of the reasoning. Johan's belief in me gave me the confidence to come to MIT. I am very grateful for all of this.

During my studies at MIT, I have also had the fortune of often interacting with Shafi Goldwasser. Her fantastic courses, rapid ideas, unusual associations and amazing ability of finding connections between seemingly unrelated phenomena opened up a new world.

Alon Rosen was my first co-author. As such he suffered from my worst writing and most disorganized mumblings. He opened up his house for me, and has helped me and guided me, both as a researcher and as a friend, from that point on. I am very grateful for this.

I am also grateful to Madhu Sudan for always being available to answer technical questions. Ran Canetti took me seriously, in an early stage of my research, despite my apparent lack of knowledge. I am grateful for this and for our enjoyable breakfast meetings at Le Pain Quotidien during my visit to IBM. Tal Rabin has with her human warmth and wise opinions on both science and life, guided me in many important situations. I am very grateful to her.

I am very grateful to Abhi Shelat for our perpetual and very fruitful discussions on all topics and to Vinod Vaikunthathan for always having the time and passion to listen to my ideas. Both Abhi and Vinod have had a strong impact on the research in this thesis and are both dear friends.

Other researchers that have deeply influenced my research include Boaz Barak, Cynthia Dwork, Isaac Elias, Yuval Ishai, Adam Kalai, Hugo Krawczyk, Yehuda Lindell, Moni Naor, Yael Tauman-Kalai and Gustav Hast. I am very grateful for my discussions with them.

My experience at MIT also owes much to Adi Akavia, Ben Adida, Susan Hohenberger, Guy Rothblum, Chris Peikert and Steve Weis.

My understand of the topics in this thesis have benefited a lot from discussions with friends. I am grateful to Marcus Better, Omar Berrada, Ulrik Dahlerus, Crispin Dickson, Magnus Larsson, Madelaine Levy, Johan Liljefors, Jacob Norda, Daniel Lovas, Andreas Röberg, Sandra Virag, Rickard Wärnelid, Magnus Åslund, and in particular Sasha Devore, whose comments, advice and support has had an important impact on this thesis.

Finally, my parents JULIA ROMANOWSKA and NATAN PASS and sisters ARIELLA and ESTHER-LOU.

Please forgive me any omissions.

# Abstract

The seminal work of Goldwasser, Micali and Rackoff put forward a computational approach to knowledge in interactive systems, providing the foundation of modern Cryptography. Their notion bounds the knowledge of a player in terms of his *potential* computational power (technically defined as polynomial-time computation).

In this thesis, we put forward a stronger notion that *precisely* bounds the knowledge gained by a player in an interaction in terms of the *actual* computation he has performed (which can be considerably less than any arbitrary polynomial-time computation).

Our approach not only remains valid even if $\mathcal{P} = \mathcal{NP}$, but is most meaningful when modeling knowledge of computationally easy properties. As such, it broadens the applicability of Cryptography and weakens the complexity theoretic assumptions on which Cryptography can be based.

# 1 Introduction

Information is not knowledge.

Albert Einstein

The seminal works of Goldwasser and Micali [31] and of Goldwasser, Micali and Rackoff [32] put forward a computational approach to knowledge in interactive systems, providing the foundation of modern Cryptography. 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 thesis, 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). For convenience, we do so in the context of interactive proofs [32], but it should be appreciated that our notion extends to more general types of interactions—in particular, secure encryption schemes and general secure multi-party (i.e., distributed) computations.

## 1.1 Zero-Knowledge Proofs

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali and Rackoff [32] 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. For instance, a zero-knowledge proof can be used to convince a verifier that a certain (large) number $N = pq$ is the product of two primes, without revealing the actual factors $p$ and $q$.

Goldwasser, Micali and Rackoff's definition [33] 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!*

## A Stronger Desiderata

We wish to put forward a notion of zero knowledge that guarantees that also all *individual* verifiers (rather than the class of polynomial-time verifiers) learn nothing new. A first step towards this goal was taken already in [29]. The refinement of [29, 27], *aimed at measuring the "actual security"* of a zero-knowledge proof system, calls for a tighter coupling between the *worst-case* running-time of $V$ (i.e., an upperbound on the running-time of $V$ in any interaction) and the expected running-time of $S$: a proof is zero knowledge with tightness $t(\cdot)$ if there exists a fixed polynomial $p(\cdot)$ such that the expected running time of $S(x)$ is upper-bounded by $t(|x|)$ times the worst-case running time of $V(x)$ plus $p(|x|)$. In essence,

*The tightness of a zero-knowledge proof bounds the knowledge of the Verifier in terms of the* ***upper-bound*** *on its running-time (in any interaction).*

Let us argue, however, that such coupling may still be insufficient, even when the tightness function is a constant and the polynomial $p(\cdot)$ is identically 0.

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 $\Omega(n^{50})$. 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

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

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])

This discussion shows that we need a stronger notion of zero knowledge. This is what we put forward in this thesis: 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*). In essence, whereas prior notions of zero knowledge were content with requiring a simulation that only incurs a small slow down relative to the the worst-case running time of the verifier, our notion calls for a small, *precise* slow down with respect to the actual running time in each *particular* execution. Thus,

> *We bound the knowledge of the Verifier in terms of its* ***actual*** *computation.*

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 $t(n)$ also has precision $t(n)$. 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.

## Stronger Implications

The original definition of Zero Knowledge only considers polynomial-time verifiers. In contrast, our definition does not make any reference to complexity classes; our verifiers may well be exponential-time, or take exponentially many steps a substantial fraction of the time, without trivializing the notion of zero knowledge. This not only makes our definition more general and conceptually simpler, but also more applicable. Let us give some examples.

*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

[2]Consider for instance, the protocol of Feige-Shamir [24] when instantiated with Goldreich, Micali and Wigderson's Graph 3-Coloring proof system [29]. 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 $\Omega(n^{50})$.

interacting with the Prover) in expected time that is comparable to the worst-case running-time of the Verifier. Barak and Lindell [6] 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. In contrast, traditional zero-knowledge proofs with a strict polynomial-time simulator having optimal tightness (i.e., the worst-case running-time of the simulator is equal to the worst-case running-time of the verifier) only guarantee that the property can be computed with (roughly) the same success probability but using a running-time that potentially *always* is equal to the worst-case time of $V$.

*Securing Semi-Easy Properties.* Whereas standard Zero-Knowledge proofs provide the guarantee that hard properties of the statement proved remain secret, Precise Zero Knowledge furthermore guarantees that the computation of also "semi-easy" ones is not *facilitated.* In many natural situations this may be very important. Consider, for instance, data mining. A large database may very well be public, but the relations among its records may not be automatically evident. One researcher may have found a correlation between —say— race and wealth in $n^5$ work, but may want to prove his discovery in zero knowledge in order to preserve the remaining privacy of the record holders to the maximum possible extent. In particular, we would like to guarantee that if a verifier $V$ can compute a second relation — say between, race and a given disease — with a certain success probability and a running time that is specified by some probability distribution, then this relation can be discovered with (roughly) the same success probability and running-time distribution, without the help of the prover; in other words, we wish to preserve the success/time-distribution of a verifier. (Note that this is significantly stronger and more meaningful than simply requiring that the relation can be computed in time comparable to the worst-case running-time of $V$, as the worst-case running-time of the verifier might potentially be very large.[3] This is exactly what Precise Zero Knowledge stipulates and guarantees. In essence, precise zero knowledge proofs leave intact the security of not only hard properties, but also semi-easy ones.[4] Consequently, precise zero-knowledge provides meaningfulness to the notion of a zero-knowledge proof also for languages in $\mathcal{P}$ (just as in the above data mining example).[5] That is, they keep zero knowledge very meaningful *even if* $\mathcal{P} = \mathcal{NP}$!

*"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 [25, 14, 19]. 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.

[3]Consider, for instance, a verifier that has relatively feeble resources. Such a verifier could nevertheless buy a lottery ticket that with some small probability could give him access to a large amount of computational resources.

[4]For a more "mathematical" example of the same notion, consider a prover that wishes to prove that a certain graph is three-colorable, but without facilitating the computation of any other (potentially easy) property of the graph (such as for instance its maximum matching).

[5]Note that it is easy to construct "trivial" precise zero-knowledge proofs for all languages in $\mathcal{P}$: the proof system consisting of the prover sending nothing and the verifier checking itself the validity of the statement, is clearly zero-knowledge with linear precision. The main challenge is, however, to construct zero-knowledge proofs with linear precision, where the running time of the honest verifier is (significantly) smaller than the time needed to decide the language.

## 1.2 Proofs of Knowledge

The notion of a proof of knowledge was intuitively introduced by Goldwasser, Micali and Rackoff [32] and was formalized by Feige, Fiat and Shamir [22] and Tompa and Woll [57]. Loosely speaking, an interactive proof of $x \in L$ is a proof of knowledge if the prover convinces the verifier that it *possesses*, or can *feasibly compute*, a witness $w$ for the statement $x$. Whereas any $\mathcal{NP}$-witness for the statement $x \in L$ trivially is a proof of knowledge of $x \in L$, zero-knowledge interactive proofs (or other general types of interactive proofs) are not necessarily proofs of knowledge.

Again, as "feasible" computation is defined through the notion of probabilistic polynomial-time, the notion of a proof of knowledge is essentially formalized as follows: an interactive proof of $x \in L$ is a proof of knowledge if there exists a probabilistic polynomial-time *extractor* machine $E$, such that for any probabilistic polynomial-time prover $P$, $E$ on input the description of $P$ and any statement $x \in L$ readily outputs a valid witness for $x \in L$ if $P$ succeeds in convincing the Verifier that $x \in L$.

The rational behind this definition is that any $P$ that convinces the verifier, can execute the extractor $E$ on its own code and therefore reconstruct the witness in probabilistic polynomial-time. Thus, in a proof of knowledge

*$P$ will only succeed in convincing the verifier that $x \in L$ if $P$ in probabilistic polynomial-time can compute a witness for $x \in L$.*

Halevi and Micali [36] 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 that of $P$. Thus,

*$P$ 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 to $P$'s **expected** running-time.*

We notice that the same arguments as used in the previous section also apply to show that even the stronger notion of a conservative proof of knowledge only "loosely" bounds the knowledge of $P$: Consider a prover $P$ 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 expected running-time of $P$ is $\Omega(n^{50})$ and thus even a conservative proof of knowledge with "optimal" tightness only requires that $P$ can compute a witness for $x$ in $\Omega(n^{50})$ steps (whereas $P$ most of the time only takes $n$ steps)

In order to more precisely bound the knowledge of the prover, we put forward the notion of a proof of knowledge which instead provides the guarantee that

*$P$ will only succeed in convincing the verifier that $x \in L$ if $P$ can compute a witness for $x \in L$ in same time (within, say, a constant factor) as 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. We call a proof satisfying the above property a *precise proof of knowledge* (or a proof of knowledge with linear precision).

Note that in contrast to traditional notions of proofs of knowledge, 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.

Furthermore, 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.)

## 1.3 Overview of the Thesis

The main contribution of this thesis is the introduction of precise definitions of zero-knowledge and proofs of knowledge. We also investigate the existence of zero-knowledge proofs satisfying our stronger definitions. In this journey, we additionally construct non-trivial proof of knowledge protocols satisfying our stronger notions.

**Chapter 2 - Preliminaries.** We introduce basic notation and recall basic notions that will be used throughout the thesis.

**Chapter 3 - Classical Work on Zero-Knowledge.** We recall classical definitions of a zero-knowledge proof, a proof of knowledge, and a witness indistinguishable proof (which is a weakening of the notion of a zero-knowledge proof). This chapter also contains descriptions of some of the major results on zero-knowledge proofs, as well as brief descriptions of some classical zero-knowledge protocols.

**Chapter 4 - Capturing Knowledge in Expectation.** We introduce new definitions of zero-knowledge and proofs of knowledge, which bound the knowledge of a player in terms of his *expected* running-time, or in terms of *higher-order moments* of his running-time. For instance, a zero-knowledge proof with *expected (or $i$'th moment) precision* requires that the expectation ($i$'th moment) of the running-time distribution of the simulator is closely related to the expectation ($i$'th moment) of the running time of the verifier. Although these notions are considerably weaker than our precise notions of zero knowledge and proofs of knowledge, they are already a significant strengthening of the traditional notions and are interesting in their own right – the notion of expected precise zero-knowledge guarantees that the *average* amount of resources needed by a verifier to perform some task is not significantly altered by the zero-knowledge proof; the notion of a higher-moment precise zero-knowledge proof guarantees the same thing, even if the computation-cost function (i.e., the cost per computation step) of the verifier is non-linear.[6]

We furthermore investigate to what extent known zero-knowledge protocols and simulators satisfy our new notions of expected precise and higher-moment precise zero-knowledge.

**Chapter 5 - Capturing Knowledge Precisely.** We introduce our notions of precise zero-knowledge and precise proofs of knowledge. We additionally introduce a potentially weaker notion of a precise proof of knowledge, called an emulatable precise proof of knowledge, which is particularly useful in cryptographic applications (and most notably in our constructions).

This chapter also discusses basic closure properties of precise zero knowledge and contain a formalization of the statement that a precise zero-knowledge proof guarantees that the running-time distribution of the simulator is (roughly) the same as the running-time distribution of the verifier.

**Chapter 6 - Existence of Precise Zero Knowledge.** We investigate the existence of precise zero-knowledge proofs. It should be appreciated that without any trusted set-up assumptions, none of the known zero-knowledge protocols and simulators satisfy our precise simulation requirement. In particular, we show that only "trivial" languages have precise zero-knowledge proof systems with so called *black-box simulators* (this is a severe lower-bound as all classical zero-knowledge protocols, and all "practical" zero-knowledge protocols indeed have black-box simulators). We, however, manage to prove the existence of precise zero-knowledge protocols in a variety of settings under standard complexity assumptions. To prove our positive results, while avoiding our impossibility ones, we rely on a simulation that *learns* (in a very weak sense) the running-time distribution of the verifier, and thereafter *exploits* it in order to perform a precise simulation.

ACKNOWLEDGMENTS. This thesis is based on joint work with Silvio Micali and has been supported by NSF Grant 0326277 and an MIT Akamai Presidential Fellowship. Portions of the thesis originally appeared as an extended abstract in the *38'th Symposium on Theory of Computing* [44].

[6]As already mentioned, for the case of proofs of knowledge, the notion of a conservative proof of knowledge [36] also bounds the knowledge of a player in terms of its expected running-time (but not in terms of higher-order moments of it).

## 1.4 A Wi(l)der Perspective

In this final section of the introduction we permit ourselves to put the results of this thesis in a wider context.

PRECISE CRYPTOGRAPHY. We emphasize that whereas the treatment in this thesis focuses of zero-knowledge *proofs*, the notion of precise zero knowledge can be applied also to more general cryptographic protocols, e.g., encryptions [31] and secure computations [29]. For concreteness, consider encryption. Recall that Goldwasser and Micali's notion of security for encryption schemes requires that a ciphertext does not reveal more than what can be computed in probabilistic polynomial-time given only the length of the encrypted plain-text [31]. A precise variant of their notion would instead be to require that anything that can be computed in time $t$ given the cipher-text can be computed in time, say, $2t$, given just the length of the plain text. Although a full investigation of precisely secure encryptions schemes is outside the scope of this thesis we mention that any secure encryptions scheme with *pseudo-random* ciphertexts (i.e., any encryption scheme having the property that the encryption of a message looks "random") is a precisely secure encryption scheme. (Interestingly, this gives a theoretical foundation to the folklore belief that a "good" encryption scheme should "scramble" a message and make it look random).

THE IMPORTANCE OF KNOWLEDGE. The notion of knowledge has grown more and more important in mathematical sciences such as Computer Science (and in particular Cryptography, Distributed Computing and Artificial Intelligence), Game Theory and Epistemic Logic. We believe that obtaining a fuller understanding of this fundamental notion might not only lead to major advances in all the above fields, but also bears the potential of actually bringing these fields closer together. In particular, we believe that our notions of knowledge provide a useful framework for modeling *bounded-rational* agents in Game Theory; such an approach could make it possible to apply game theoretic reasoning also to cryptographic protocols.

KNOWLEDGE AND COMPUTATION. Computation and reasoning is at heart of the notion of knowledge. We thus believe that any modern theory of knowledge *needs* to take computation into account. Of course, "time" is just one aspect of computation, and indeed all of our notions (as well as protocols) extend to also consider other parameters of a computation (e.g., space-complexity).

A PRECISE APPROACH TO COGNITIVE NOTIONS. In this thesis we have "only" addressed the notion of knowledge. The field of Game Theory allows for mathematical formalizations of other cognitive notions such as *beliefs*, *rationality* and *preferences*. Currently "main-stream" Game Theory defines these notions in terms of the *expectations* of certain utility functions for the players (much like our notions of expected precise zero knowledge and proofs of knowledge). It is well-known in the Game Theory literature that such an approach might be inadequate in several situations: For instance, a player might very well prefer to be killed with probability one in a trillion (e.g., taking an airplane), to being robbed with 10% probability, although both options have the same expected utility. Much like our precise notion of knowledge, one could consider extending also game theoretic definitions to take into account the actual *distribution* of the utility of players.

Information is not knowledge. Knowledge is not wisdom. Wisdom is not truth. Truth is not beauty. Beauty is not love. Love is not music. Music is the best.

Frank Zappa

# 2 Preliminaries

## 2.1 Basic Notation

### 2.1.1 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 $\alpha$ is a binary string, then $|\alpha|$ denotes $\alpha$'s length and $\alpha_1 \cdots \alpha_i$ denotes $\alpha$'s $i$-bit prefix.

PROBABILISTIC NOTATION. We employ the following probabilistic notation from [34]. We focus on probability distributions $X : S \rightarrow R^+$ over finite sets $S$.

*Probabilistic assignments.* If $D$ is a probability distribution and $p$ a predicate, then "$x \overset{R}{\leftarrow} D$" denotes the elementary procedure consisting of choosing an element $x$ at random according to $D$ and returning $x$, and "$x \overset{R}{\leftarrow} D \mid p(x)$" 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 $\Pr[x_1 \overset{R}{\leftarrow} D_1;\ x_2 \overset{R}{\leftarrow} D_2;\ \ldots\ :\ p(x_1, x_2, \ldots)]$ denotes the probability that $p(x_1, x_2, \ldots)$ will be true after the ordered execution of the probabilistic assignments $x_1 \overset{R}{\leftarrow} D_1;\ x_2 \overset{R}{\leftarrow} D_1;\ \ldots$

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

*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 "short-cut": 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 $\nu(n) < n^{-c}$.

## 2.1.2 Protocol Notation

We assume familiarity with the basic notions of an *Interactive Turing Machine* [33] (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 $1, 2, \ldots$. 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 thesis 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 $M_A = \{m_A^1, m_A^2, \ldots\}$, $M_B = \{m_B^1, m_B^2, \ldots\}$ 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 received by $A$ and in $m_B^i$ being the $i$'th message received by $B$. 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 $i$th 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.

## 2.2 Basic Notions

### 2.2.1 Basic Complexity Classes

We recall the definitions of the basic complexity classes $\mathcal{P}, \mathcal{NP}$ and $\mathcal{BPP}$.

THE COMPLEXITY CLASS $\mathcal{P}$. We start by recalling the definition of the class $\mathcal{P}$, i.e., the class of languages that can be decided in (deterministic) polynomial-time.

*Definition* 1 *(Complexity Class $\mathcal{P}$).* A language $L$ is *recognizable in (deterministic) polynomial-time* if there exists a deterministic polynomial-time algorithm $M$ such that $M(x) = 1$ if and only if $x \in L$. $\mathcal{P}$ is the class of languages recognizable in polynomial time.

THE COMPLEXITY CLASS $\mathcal{NP}$. We recall the class $\mathcal{NP}$, i.e., the class of languages for which there exists a proof of membership that can be verified in polynomial-time.

*Definition* 2 *(Complexity Class $\mathcal{NP}$).* A language $L$ is in $\mathcal{NP}$ if there exists a Boolean relation $R_L \subseteq \{0, 1\}^* \times \{0, 1\}^*$ and a polynomial $p(\cdot)$ such that $R_L$ is recognizable in polynomial-time, and $x \in L$ if and only if there exists a string $y \in \{0, 1\}^*$ such that $|y| \le p(|x|)$ and $(x, y) \in R_L$.

The relation $R_L$ is called a *witness relation* for $L$. We say that $y$ is a witness for the membership $x \in L$ if $(x,y) \in R_L$. We will also let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e.,

$$R_L(x) = \{y : (x,y) \in L\}$$

We let co-$\mathcal{NP}$ denote the complement of the class $\mathcal{NP}$, i.e., a language $L$ is in co-$\mathcal{NP}$ if the complement to $L$ is in $\mathcal{NP}$.

THE COMPLEXITY CLASS $\mathcal{BPP}$. We recall the class $\mathcal{BPP}$, i.e., the class of languages that can be decided in *probabilistic* polynomial-time (with two-sided error).

*Definition* 3 *(Complexity Class $\mathcal{BPP}$).* A language $L$ is *recognizable in probabilistic polynomial-time* if there exists a probabilistic polynomial-time algorithm $M$ such that

- $\forall x \in L, \Pr[M_\bullet(x) = 1] \geq 2/3$
- $\forall x \notin L, \Pr[M_\bullet(x) = 0] \geq 2/3$

$\mathcal{BPP}$ is the class of languages recognizable in probabilistic polynomial time.

### 2.2.2 Indistinguishability

The following definition of (computational) indistinguishability originates in the seminal paper of Goldwasser and Micali [31].

*Definition* 4 *(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 $D$.

### 2.2.3 Interactive Proofs and Arguments

We state the standard definitions of interactive proofs (introduced by Goldwasser, Micali and Rackoff [33]) and arguments (introduced by Brassard, Chaum and Crepeau [11]).

*Definition* 5 *(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 5 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. [7]).

THE COMPLEXITY OF INTERACTIVE PROOFS. Whereas all of $\mathcal{NP}$ trivially has an interactive proof (where the prover simply sends the NP-witness, and the verifier checks if it is valid or not), the class of languages having interactive proof, called $\mathcal{IP}$, possibly extends beyond $\mathcal{NP}$. In particular, Shamir showed that all of $\mathcal{PSPACE}$ (i.e., all languages that can be decided in polynomial space) have interactive proofs [56]. We provide an example of a language which is not known to be in $\mathcal{NP}$ but still has an interactive proof.

**Example 27.1** Consider the language of non-isomorphic graphs of the same size (this languages is clearly in co-$\mathcal{NP}$, but is not know to be in $\mathcal{NP}$). Goldreich, Micali and Wigderson demonstrated that this language has an interactive proof [29]. Their protocol (which has soundness error $1/2$) is depicted in Figure 2.1.

PROTOCOL *GMW-GraphNonIso*

**Common Input:** an instance $G_0, G_1$ of the language GRAPHNONISO.

V uniformly chooses a bit $i$ and let $H$ be a random isomorphic copy of $G_i$.
V $\rightarrow$ P: $H$.
P $\rightarrow$ V: The bit $i'$ such that $H$ is isomorphic to $G_{i'}$.
V accepts if $i' = i$.

Figure 2.1: An interactive proof for GRAPHNONISO

The intuition behind the protocol is as follows. If the graphs $G_0, G_1$ indeed are not isomorphic, the honest prover $P$ will always be able to tell which of the graphs $H$

is isomorphic with. The protocol is thus complete (in fact, it has 0 completeness error.) On the other hand, if $G_0, G_1$ are isomorphic, then $H$ is isomorphic to both $G_0$ and $G_1$. This means that even an all powerful malicious prover will not be able to guess the bit $i$ with probability higher than $1/2$.

ARTHUR-MERLIN PROTOCOLS. In certain applications of interactive proofs/ arguments it is desirable that the verifier only uses *public* random coins (i.e., all its random coins are also revealed the prover; such proofs/arguments are called *public-coin* or Arthur-Merlin[4]. Due to the fact that the verifier in a public-coin protocol only uses public random coins, we can without loss of generality only consider public-coin proofs/arguments having the following canonical structure:

1. the verifier only sends random messages to the prover,
2. at the end of the interaction, the verifier determines whether to accept or not by applying a *deterministic* predicate to the transcript of all messages in the interaction.

In Section 3.2 we provide examples of public-coin interactive proof systems.

INTERACTIVE PROOFS WITH EFFICIENT PROVERS. For cryptographic applications it is necessary that the prover strategy can be implemented efficiently when given a witness.

*Definition* 6 *(Efficient Provers).* 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 $(P, V)$ has an *efficient prover* if $P$ is a probabilistic polynomial-time algorithm and the completeness condition of Definition 5 holds for every $x \in L$ and every $y \in R_L(x)$.

**Example 28.1** As an example of an interactive proof system with an efficient prover, consider the following "trivial" interactive proof system for a language $L \in \mathcal{NP}$, with witness relation $R_L$: on common input $x$, and auxiliary input $w \in R_L(x)$, the prover $P$ simply send the witness $w$ to verifier $V$. $V$ accepts if and only if $(x, w) \in R_L$.

In section 3.2 we provide less "trivial" examples of interactive proof systems that have efficient prover strategies.

## 2.2.4 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 7 (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 [27].

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 [27]). 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 [48, 38]. Perfectly-hiding commitment schemes can be constructed from any one-way permutation [49]. However, *constant-round* schemes are only known to exist under stronger assumptions; specifically, assuming the existence of a collection of certified clawfree permutations [28] (see also [27], Section 4.8.2.3). Constant-round statistically-hiding commitments can be constructed under the potentially weaker assumption of collision-resistant hash functions [18, 37].

# 3 Classical Work on Zero Knowledge

## 3.1 Classical Zero Knowledge Notions

### 3.1.1 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 [32, 33]. We present their definition below.[1]

*Definition* 8 *($\mathcal{ZK}$).* 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

[1]The definition we present here appears in the journal version [33] or [32]. It differs from the original definition of [32] 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 [33] mention that this refinement was independently suggested by the Oren [50], Tompa and Woll [57] and the authors.

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 [27])

*Definition* 9 *(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.

KNOWLEDGE TIGHTNESS. Goldreich, Micali and Wigderson [29], and more recently [27] 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 [27].[2] More precisely,

[2]To be precise, the authors of [29] 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 [29] 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 [27], 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.

*Definition* 10. 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 8) 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. (Looking ahead, we will introduce notions that require tightness with respect to either the *expected* running-time of the verifier, or its *actual* running-time.)

## 3.1.2 Witness Indistinguishability

The notion of Witness Indistinguishability ($\mathcal{WI}$) was introduced by Feige and Shamir in [23] 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., [24, 21, 53, 2].

We proceed to a formal definition (following [27]),

*Definition* 11 *(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 $L$.

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

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.

### 3.1.3 Proofs of Knowledge

The notion of a proof of knowledge was intuitively introduced in the paper by Goldwasser, Micali and Rackoff [32] and was formalized by Feige, Fiat and Shamir [22] and Tompa and Woll [57]. The definition was further refined by Bellare and Goldreich [5]. 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 [20] (which in turn follows that of Tompa and Woll [57]).

*Definition* 12 *(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 $z \in \{0,1\}^*$,

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

CONSERVATIVE PROOFS OF KNOWLEDGE. Micali and Halevi propose a strengthening of the definition of proofs of knowledge [36]. Their motivation is to provide a definition of proofs of knowledge which implies that the *expected* running-time of the extractor is polynomially related to the *expected* running-time of the prover in a real interaction with the honest verifier (this property is called *prover feasibility*). They note that even an extractor that makes only a polynomial number of queries to the prover could potentially "lure" the prover to take exponentially many steps (by for instance feeding him some special queries) whereas the prover would never (or almost never) take this many steps in a true interaction with a honest verifier. To circumvent this problem, they strengthen the definition of a proof of knowledge by adding the requirement that the extractor only feeds messages to the prover according to the *same* distribution as those sent by the true verifier (this is called a *valid* distribution). They show that this additional restriction (which anyway is satisfied by all traditional extractors) guarantees prover feasibility, i.e., that the expected running-time of the extraction is polynomially related to the expected running-time of the prover in an interaction with the honest verifier.

## 3.2 Classical Zero Knowledge Protocols

In this section we review some classical results on $\mathcal{ZK}$.

### 3.2.1 $\mathcal{ZK}$ proofs for all of $\mathcal{NP}$

#### The Protocol of GMW

Goldreich, Micali and Wigderson demonstrate the generality of $\mathcal{ZK}$ by showing that every language in $\mathcal{NP}$ has a zero-knowledge proof system [29].

*Theorem* 1 *([29]).* Assuming the existence of statistically binding commitments, every language in $\mathcal{NP}$ has an (efficient-prover) computational $\mathcal{ZK}$ proof system.

They prove the above theorem by demonstrating a $\mathcal{ZK}$ proof system for the $\mathcal{NP}$-language Graph Three-Coloring (GRAPH3COL) — i.e., the language consisting of graphs that can be colored using at most three colors. Since GRAPH3COL is $\mathcal{NP}$-Complete this proof system can be used for any language $L$ in $\mathcal{NP}$ – given an instance $x \in L$ simply reduce $x$ to an instance $x' \in$ GRAPH3COL and execute the $\mathcal{ZK}$ proof system for GRAPH3COL on common input $x'$. Their protocol is depicted in Figure 3.1.

We provide a sketch of why this protocol is complete, sound and zero-knowledge. Completeness follows directly, since when the prover commits to a valid coloring, the verifier will always accepts. On the other hand, if the graph is not 3-colorable, then any coloring that a malicious prover can commit to must contain two adjacent vertices with the same colors. It follows that with probability $\frac{1}{|E_G|}$, the verifier will pick this edge and the malicious prover will be caught. We conclude that the protocol has soundness error $1 - \frac{1}{|E_G|}$. The soundness error can then be decreased to become negligibly small by sequential repetition of the protocol (i.e., by letting the Prover and Verifier engage in multiple consecutive executions of the protocol.)

The zero-knowledge property intuitively follows from the fact that the verifier only "sees" two random colors, which is something that he could have generated himself. The actual proof of this seemingly simple statement turns out to be quite subtle. We construct a black-box simulator $S$ that given oracle access to any malicious verifier $V'$ outputs a view that is indistinguishable from the view of $V'$ in a true interaction with a prover. $S$ proceeds as follows:

PROTOCOL *GMW-Graph3Col*

**Common Input:** a directed graph $G = (V_G, E_G)$, with $n = |V_G|$

**Auxiliary Input for the prover:** a 3-coloring of $G$, $\bar{c} = c_0, c_1, .., c_n \in \{1,2,3\}$.

P uniformly chooses a permutation $\pi$ over 1,2,3.

P $\rightarrow$ V: Commits to $\pi(c_0), \pi(c_1), .., \pi(c_n)$ using a statistically binding commitment scheme.

V $\rightarrow$ P: Uniformly selects an edge $(i,j) \in E_G$.

P $\rightarrow$ V: Reveals $c_i, c_j$.

V accepts if and only if P correctly revealed to $c_i, c_j$ and if $c_i, c_j$ are different colors.

Figure 3.1: GMW's $\mathcal{ZK}$ Proof for GRAPH3COL

1. $S$ uniformly picks a random edge $(i,j)$ in the graph.
2. $S$ then picks a coloring $\bar{c}$ such that $c_k = 1$ for all $k \neq i,j$ and $c_i$, $c_j$ are two random, but different, colors.
3. It commits to $\bar{c}$ and feeds this commitment to $V'$.
4. If $V'$ asks for the edge $(i,j)$, $S$ reveals the commitments and outputs the generated view.
5. Otherwise, $S$ *restarts* $V'$ and repeats the above procedure.

It follows, due to the hiding property of all unrevealed commitments, that when $S$ indeed is able to output a view, this view will be indistinguishable from the view of a real interaction, Furthermore, it follows, again from the hiding property of the commitments, that the probability that $V'$ picks the edge $(i,j)$ is roughly $\frac{1}{|E_G|}$. We conclude that the expected running-time of $S$ will be $poly(|E_G|)$. (We warn the reader that a complete proof of the correctness of the above simulation is significantly more complicated and requires overcoming several subtle complications.)

**The protocol of Blum**

Blum subsequently provided a $\mathcal{ZK}$ proof system for the $\mathcal{NP}$-Complete language Hamiltonian-Cycle (HAMCYCLE) — i.e., the language consisting of all graphs containing a Hamiltonian cycle [9]. As this protocol will be used extensively in this thesis we recall it in Figure 3.2.

The analysis of the protocol of Blum is very similar to that of the protocol of GMW. The main advantage of this protocol is its strong soundness guarantee:

PROTOCOL *Blum-HCProof*

**Common Input:** a graph $G$ which is part of the language HAMCYCLE.

**Auxiliary Input for Prover:** a Hamilton cycle C in the graph $G = (V, E)$.

V uniformly chooses a random permutation $\pi$ of the vertices $V$ and commits to the adjacency matrix of resulting permuted graph. That is, $V$ uses a statistically binding commitment COM to send an $|V| \times |V|$ matrix of commitments so that the $(\pi(i), \pi(j))$ entry is a commitment to 1 if $(i, j) \in E$ and is a commitment to 0 otherwise.

V $\rightarrow$ P: a uniformly chosen bit $\sigma$.

P $\rightarrow$ V:

1. If $\sigma = 0$, $P$ sends $\pi$, and also reveals the full matrix of commitments sent.
2. If $i = 1$, $P$ instead only reveals a cycle in the committed permuted graph, i.e., only reveals the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C$.

V proceeds as follows.

1. If $i = 0$ it checks that $P$ properly decommits, and that the revealed graph is isomorphic to $G$, via $\pi$.
2. If $i = 1$, it checks that $P$ properly decommits, that all the revealed values are 1, and that the corresponding entries form a simple cycle.

Figure 3.2: Blum's $\mathcal{ZK}$ Proof for $\mathcal{NP}$

The protocol resulting from only a single invocation of Blum's protocol is a zero-knowledge interactive proof with soundness error only 1/2 (in contrast, recall that a single invocation of GMW's protocols has soundness error $1 - \frac{1}{|E_G|}$).

### 3.2.2 $\mathcal{ZK}$ proofs for all of IP

Ben-Or, Goldreich, Goldwasser, Håstad, Kilian, Micali and Rogaway [8] show that the result of Goldreich, Micali and Wigderson can be extended to show the following:

*Theorem* 2 *([8]).* Every language in $\mathcal{IP}$ (i.e., every language that has an interactive proof system) has a computational $\mathcal{ZK}$ proof system.

We provide an outline of their proof. A fundamental result by Goldwasser and Sipser [35] shows that any language having an interactive proof also has a public-coin interactive proof. Thus, it is sufficient to show that any language having a public-coin interactive proof system also has a $\mathcal{ZK}$ proof. Given a public-coin interactive proof system $(P, V)$ for a language $L$, we construct a (computational)

$\mathcal{ZK}$ proof system $(P_2, V_2)$. $(P_2, V_2)$ is defined as follows. On input $x \in L$, $P_2$ and $V_2$ proceed in two stages.

1. In the first stage, $P_2$ and $V_2$ execute the original public-coin proof system $(P, V)$ on common input $x$, *but* with the exception that instead of sending the messages of $P$ in the clear, $P_2$ rather sends *commitments* (using a statistically binding commitment scheme) to these messages. Note that this change does not affect the strategy of $V$, since $V$ only sends random strings as its messages.

2. In the second stage, $P_2$ uses GMW's proof system to prove the $\mathcal{NP}$-statement that "the messages committed to by $P_2$ (in the first phase of the protocol) would make $V$ accept". Note that is indeed an $\mathcal{NP}$-statement, since 1) the verifier $V$ only sends public-coins, and 2) $V$ is a polynomial-time algorithm.

### 3.2.3 Statistical $\mathcal{ZK}$ Arguments for $\mathcal{NP}$

Brassard, Chaum and Crepeau [11] demonstrate the existence of a *perfect/ statistical $\mathcal{ZK}$ argument* for $\mathcal{NP}$. In fact, it can be shown that if instantiating the commitment scheme COM in Protocols *GMW-Graph3Col* and *Blum-HC* with a perfectly-hiding (statistically-hiding) commitment scheme, then both protocols are in fact a perfect (statistical) $\mathcal{ZK}$ arguments for $\mathcal{NP}$.

*Theorem* 3 *([11]).* Assuming the existence of perfectly/statistically hiding commitments, every language in $\mathcal{NP}$ has an (efficient-prover) perfect/statistical $\mathcal{ZK}$ argument system.

### 3.2.4 On the Existence of Statistical $\mathcal{ZK}$ Proofs

We have above presented two complimentary results. The result of Goldreich, Micali and Wigderson (GMW) concerns interactive *proofs* (as opposed to arguments) satisfying *computational* zero-knowledge, whereas the result of Brassard, Chaum and Crepeau (BCC) concerns the weaker notion of interactive *arguments*, while achieving the stronger notion of *perfect/statistical* zero-knowledge. In other words, the protocol of GMW is secure against an unbounded prover, but a computationally bounded verifier, whereas the protocol of BCC is secure against an unbounded verifier, but a computationally bounded prover.

It is unknown whether all of $\mathcal{NP}$ has statistical $\mathcal{ZK}$ proofs (i.e., protocols that are secure against both unbounded provers and verifiers). Indeed, Aiello and Håstad show that the existence of such a proof system would imply the collapse of the Polynomial Hierarchy [1].

Nevertheless, although it is unlikely that all of $\mathcal{NP}$ has statistical $\mathcal{ZK}$ proofs several interesting languages (including for instance GRAPHNONISO mentioned in Section 2.2.3 which are not known to be in $\mathcal{P}$ have statistical $\mathcal{ZK}$ proofs. Furthermore, the class of languages having such proof systems, denoted **SZK**, has interesting complexity theoretic properties, including complete-problems, and closure under complement [55]. We refer the interested reader to the thesis of Vadhan for an extensive study of this topic [58].

### 3.2.5 Round-efficiency of Zero-knowledge Proofs

Recall that the original $\mathcal{ZK}$ protocols for $\mathcal{NP}$ of [29, 9, 11] require repeating an atomic protocol sequentially (at least) $n$ times to obtain a $\mathcal{ZK}$ proof (argument) with soundness error $2^{-n}$. As such, the round-complexity of these protocols grows as a function of the soundness error: the smaller the error, the larger the number of rounds.

More recent results by Feige and Shamir [24] and Goldreich and Kahan [28] instead show how to obtain *constant-round* $\mathcal{ZK}$ protocols for $\mathcal{NP}$ with negligible soundness error. On a high-level (and over simplified), these protocols rely on an appropriately adjusted *parallel* repetition of the protocols of [29, 9, 11]. (We mention that it is unknown whether the protocols obtained by a simple parallel repetition of the protocols of [29, 9, 11] are $\mathcal{ZK}$.)

*Theorem* 4 *([24]).* Assuming the existence of constant-round perfectly-/statistically-hiding commitments, every language in $\mathcal{NP}$ has an constant-round (efficient-prover) perfect/statistical $\mathcal{ZK}$ argument system.

*Theorem* 5 *([24]).* Assuming the existence of one-way functions, every language in $\mathcal{NP}$ has an constant-round (efficient-prover) computational $\mathcal{ZK}$ argument system.

*Theorem* 6 *([28]).* Assuming the existence of constant-round statistically-hiding commitments, every language in $\mathcal{NP}$ has a constant-round (efficient-prover) computational $\mathcal{ZK}$ proof system.

# 4 Capturing Knowledge in Expectation

As previously mentioned, standard definitions of $\mathcal{ZK}$ and proofs of knowledge bound the knowledge of machine in terms of its the *worst-case* running-time. (As discussed in Section 3.1.3 the proof of knowledge definition of Micali and Halevi is an exception to this.)

We here put forward natural extensions of these notions, aimed at bounding the knowledge of a player in terms of its *expected* running-time, or in terms of *higher-order moments* of its running-time.

Although these notions are considerably weaker than our precise notions of zero knowledge and proofs of knowledge (which will be introduced in Chapter 5), they are already a significant strengthening of the traditional notions and are interesting in their own right – the notion of expected precise zero-knowledge guarantees that the *average* amount of resources needed by a verifier to perform some task is not significantly altered by the zero-knowledge proof; the notion of a higher-moment precise zero-knowledge proof guarantees the same thing, even if the computation-cost function (i.e., the cost per computation step) of the verifier is non-linear.

As already mentioned, for the case of proofs of knowledge, the notion of a conservative proof of knowledge [36] also bounds the knowledge of a player in terms of its expected running-time (but not in terms of higher-order moments of it). Our definition is, however, quite different: The definition of Micali and Halevi requires extraction to be performed in a *black-box* manner[1] whereas our definition makes no such restrictions; this makes our definition potentially more applicable.

[1]Furthermore, their definition puts additional restrictions on the nature of this black-box extraction. In particular, they require that the extractor feeds the prover messages according to the same distribution as the verifier.

Furthermore, arguably our definition is simpler, and has a clearer "semantics" (in particular, it easily extends to higher-order moments as well.)

In the context of $\mathcal{ZK}$ proofs, to the best of our knowledge, our definition is the first one to explicitly bound the knowledge of the verifier in terms of its expected running-time.

## 4.1 Expected Precise Zero Knowledge

*Definition* 13 *(Perfect Expected 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 $p : N \times N \to N$ a monotonically increasing function. We say that $(P,V)$ is *perfect $\mathcal{ZK}$* with *expected precision* $p$ if, for every probabilistic polynomial-time ITM $V'$, there exists a probabilistic algorithm $S$ such that the following two conditions holds:

1. The following two ensembles are identical:

   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 all $x \in L, z \in \{0,1\}^*$,

$$\mathbf{Exp}[\text{STEPS}_{S_\bullet(x,z)}] \le p(|x|, T_{x,z})$$

   where $T_{x,z}$ denotes the expected running-time of $V'_\bullet(x,z)$ in an interaction with $P_\bullet(x,y)$ for any $y \in R_L(x)$.

We refer to an algorithm $S$ as above as an *expected precise simulator*, or as a *simulator with expected precision p.*

COMPUTATIONAL/STATISTICAL $\mathcal{ZK}$. We obtain the notion of *statistically/ computationally expected precise $\mathcal{ZK}$* by requiring that the two ensembles of Condition 1 be statistically close/computationally indistinguishable over $L$.

HIGHER-ORDER MOMENT PRECISION. One can naturally generalize the above definition to consider also higher-order moments (and not just the first-order moment).[2] We obtain the notion of *$i$'th moment precise $\mathcal{ZK}$* by exchanging condition 2 in Definition 13 for the following condition:

2′. For all $x \in L, z \in \{0,1\}^*$,

$$\mathbf{Exp}[(\text{STEPS}_{S_\bullet(x,z)})^i] \le p(|x|, T^{(i)}_{x,z})$$

   where $T^{(i)}_{x,z}$ denotes the $i$'the moment of the running-time distribution of $V'_\bullet(z)$ in an interaction with $P_\bullet(y)$ on common input $x$ and any $y \in R_L(x)$.

[2]Recall that the $i$'th moment of a random variable $X$ is $\mathbf{Exp}[X^i]$.

REMARK: Note that our definition of expected (higher-moment) precise $\mathcal{ZK}$ only differs from the standard definition of $\mathcal{ZK}$ in that we additionally require that the expectation (higher-moment) of the running-time distribution of the simulator (on inputs in the language) is "close" to the expectation (higher-moment) of the running-time distribution of the verifier in true interactions with a prover.

## 4.2 Expected Precise Proofs of Knowledge

We present our definition of expected precise proofs of knowledge. Intuitively we say that $(P, V)$ is a proof of knowledge with expected 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 a random joint-view $(view_{P'}, view_V)$ of an execution between $P'$ and $V$ on common input $x$ it holds that the *expected* running-time of $E$ on input $view_{P'}$ is smaller than $p(|x|, T)$ where $T$ denotes the *expected* running-time of $P'$ in an interaction with $V'$ on common input $x$.

More precisely,

*Definition* 14 *(Expected 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 *expected 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 all $x, z \in \{0, 1\}^*$,

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

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

$$\mathbf{Exp}[(view_{P'}, view_V) \leftarrow \langle P'_\bullet(z), V_\bullet\rangle(x) : \text{STEPS}_{E_\bullet(view_{P'})}] \le p(|x|, T_{x,z})$$

where $T_{x,z}$ denote the expected running time of $P'_\bullet(x, z)$ in an interaction with $V_\bullet(x)$.

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

STATISTICALLY/COMPUTATIONALLY SOUND PRECISE PROOFS OF KNOWLEDGE. We obtain the notion of a *statistically-sound expected precise proof of knowledge* by exchanging condition 1 in Definition 14 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 expected precise proof of knowledge* by furthermore adding the restriction that $P'$ is a probabilistic polynomial-time machine.

HIGHER-ORDER MOMENT PRECISION. We obtain the notion of *$i$'th moment precise* proofs of knowledge by by exchanging condition 2 in Definition 13 for the following condition:

2′. For all $x, z \in \{0,1\}^*$,

$$\mathbf{Exp}[(view_{P'}, view_V) \leftarrow \langle P'_\bullet(z), V_\bullet\rangle(x) \ : \ (\text{STEPS}_{E_\bullet(view_{P'})})^i] \leq p(|x|, T^{(i)}_{x,y})$$

where $T^{(i)}_{x,z}$ denotes the $i$'th moment of the running time distribution of $P'_\bullet(x,z)$ in an interaction with $V_\bullet(x)$.

REMARK: We provide a brief comparison with the definition of Halevi-Micali [36]. As mentioned in Section 3.1.3, the definition of [36] requires that extraction is performed in a black-box way, and that the black-box extractor feeds the prover messages according to the *same* distribution as the honest verifier. [36] shows that this restriction implies that the running-time of the extractor is polynomially related to the *expected* running-time of the prover in a true interaction with the verifier. In our definition we make no such restrictions on the extractor. Instead, we explicitly require a close relation between the (expected) running-time of the extractor and the expected running-time of the prover.

## 4.3 Existence of Expected Precise Zero Knowledge

We show that "natural" black-box simulators have good expected precision. By natural we mean black-box simulators that only feed the verifier messages that are *indistinguishable* from true prover messages. We use the word natural, since all known black-box simulators indeed are of this type.

This result can be seen as a generalization (and strengthening) of the results of Halevi and Micali, showing a corresponding result for proofs of knowledge, if assuming that the distribution of the messages produced by the extractor is *identical* to the distribution of true verifier messages.

Our result is also related to a result by Katz and Lindell [41] showing that every proof system that has a certain strong type of natural simulator (in particular for which indistinguishability holds for super polynomial-time), is also $\mathcal{ZK}$ for *expected* polynomial-time verifiers (as opposed to strict polynomial-time verifiers). Our results rely on proof techniques from both the above-mentioned results.

We proceed to a formal definition of *natural black-box simulators*. Recall that a black-box simulator only has black-box access to the malicious verifier $V'$ and proceeds by feeding messages to $V'$ and waiting to receive back answers, much like a standard prover; furthermore the black-box simulator can rewind and restart $V'$. To simplify notation, we assume that a black-box simulator always feeds (partial) views to $V'$ containing all messages $V'$ has received from the beginning of the protocol execution; this is called a query.

*Definition* 15 *(Natural Black-box Simulators).* Let $(P, V)$ be a $k$-round black-box $\mathcal{ZK}$ proof (argument) system for the language $L \in \mathcal{NP}$ with the witness relation $R_L$, and let $S$ denote its black-box simulator. We say that $S$ is a *natural black-box simulator*, if for every probabilistic polynomial-time interactive machine $V'$, the following two ensembles, indexed by $x \in L, y \in R_L(x), z, r \in \{0,1\}^*, i \in [k(|x|)], j \in N$, are computationally indistinguishable over $L$.

- $\left\{view \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \ : \ view^i\right\}$
- $\left\{view \leftarrow \text{query}_{S_\bullet^{V'_r(x,z)}(x)}(j) \;\middle|\; |view| = i \ : \ view\right\}$

where $\text{query}_{S_{r'}^{V'_r(x,z)}(x)}(j)$ denotes the $j$'th (partial) view feed by $S_{r'}$ to $V'_r(x,z)$ (or $\perp$ if $S_{r'}$ feeds less than $j$ queries to $V'_r(x,z)$), $view^i$ denotes the partial view consisting of the first $i$ rounds in the view $view$, and $|view|$ denotes the number of communication rounds in the partial view $view$.

*Lemma* 1. Let $S$ be a natural black-box simulator for the perfect/statistical/ computational $\mathcal{ZK}$ proof systems $(P, V)$ for $L$ with witness relation $R_L$. If $S$ has expected running-time $p_1(n)$ and makes in expectation $p_2(n)$ number of queries to its oracle, then there exists a negligible function $\mu(n)$ such that $S$ has expected precision $p(n,t) = p_1(n) + p_2(n)t + \mu(n)$.

**Proof:** Consider any probabilistic polynomial-time adversary verifier $V'$. Fix generic $x \in L, w \in R_L(x), z, r \in \{0,1\}^*$. Let $E_{V'}^{real}$ denote the expected running time of $V'_r(x,z)$ in an interaction with $P_\bullet(x,w)$. We start by noting that it follows (by the linearity of expectations) that the expected running-time of $S$ *including* the steps of $V'$ is $p_1(n)$ plus the expected running-time of $V'$ in the queries posed by $S$. We proceed to bound the latter quantity, which we call $E_{V'}^{sim}$. To simplify notation, we let

$$q_\bullet(j) = \text{query}_{S_\bullet^{V'_r(x,z)}(x)}(j)$$

Let $R$ denote an upper-bound on the number of random coins used by $S_\bullet^{V'_r(x,z)}(x)(j)$. Then,

$$E_{V'}^{sim} = \frac{1}{2^R} \sum_{r \in \{0,1\}^R} \sum_j^{\infty} \left[\text{STEPS}_{V'}(q_r(j))\right] = \sum_j^{\infty} \mathbf{Exp}\left[\text{STEPS}_{V'}(q_\bullet(j))\right] \quad (4.1)$$

Expanding the above expression, we get

$$E_{V'}^{sim} = \sum_{j}^{\infty} \sum_{i}^{k(n)} \Pr\left[|q_{\bullet}(j))| = i\right] \mathbf{Exp}\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v)\right]$$

We bound the second factor

$$\begin{aligned} &\mathbf{Exp}\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v)\right] = \\ &\sum_{t}^{\infty} \Pr\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v) \geq t\right] = \\ &\sum_{t}^{\text{TIME}_{V'}(|x|)} \Pr\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v) \geq t\right] \end{aligned}$$

Since $S$ is a natural black-box simulator, and since $\text{STEPS}_{V'}$ is efficiently computable (since it is bounded by $\text{TIME}_{V'}(n)$, which in turned is bounded by a polynomial) it follows that there exists a negligible function $\mu(n)$ such that for all $t \leq \text{TIME}_{V'}(|x|)$

$$\begin{aligned} &\Bigg| \Pr\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v) \geq t\right] - \\ &\Pr\left[v \leftarrow \text{VIEW}_2[P_{\bullet}(x,y) \leftrightarrow V_r'(x,z)] \,:\, \text{STEPS}_{V'}(v^i) \geq t\right] \Bigg| \\ &= \mu(|x|) \end{aligned}$$

Since $\text{TIME}_{V'}(n)$ is polynomial in $n$, it holds that there exist some other negligible function $\mu'(n)$ such that

$$\begin{aligned} &\sum_{t}^{\text{TIME}_{V'}(|x|)} \Pr\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v) \geq t\right] - \\ &\sum_{t}^{\text{TIME}_{V'}(|x|)} \Pr\left[v \leftarrow \text{VIEW}_2[P_{\bullet}(x,y) \leftrightarrow V_r'(x,z)] \,:\, \text{STEPS}_{V'}(v^i) \geq t\right] \geq \\ &\sum_{t}^{\text{TIME}_{V'}(|x|)} \mu(|x|) = \mu'(|x|) \end{aligned}$$

In other words,

$$\begin{aligned} &\mathbf{Exp}\left[v \leftarrow q_{\bullet}(j) \,\middle|\, |v| = i \,:\, \text{STEPS}_{V'}(v)\right] - \\ &\mathbf{Exp}\left[v \leftarrow \text{VIEW}_2[P_{\bullet}(x,y) \leftrightarrow V_r'(x,z)] \,:\, \text{STEPS}_{V'}(v^i)\right] \geq \\ &\mu'(|x|) \end{aligned}$$

Since $V'$'s running time in a partial view is smaller or equal to its running-time in a full view, we get

$$\mathbf{Exp}\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \text{STEPS}_{V'}(v)\Big] \leq$$
$$\mathbf{Exp}\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \;:\; \text{STEPS}_{V'}(v)\Big] + \mu'(|x|) \leq$$
$$E^{real}_{V'} + \mu'(|x|)$$

Plugging this in to equation 4.1 we get

$$E^{sim}_{V'} \leq \sum_{j}^{\infty} \sum_{i}^{k(|x|)} \Pr\Big[|q_\bullet(j))| = i\Big]\Big(E^{real}_{V'} + \mu'(|x|)\Big) =$$
$$\Big(E^{real}_{V'} + \mu'(|x|)\Big) \sum_{j}^{\infty} \sum_{i}^{k(|x|)} \Pr\Big[|q_\bullet(j))| = i\Big] =$$
$$\Big(E^{real}_{V'} + \mu'(|x|)\Big) \sum_{j}^{\infty} \Pr\Big[q_\bullet(j)) \neq \perp\Big]$$

Letting $\#q_{r'}$ denote the number of queries made by $S$ to $V'_r(x,z)$ in an execution of $S_{r'}^{V'_r(x,z)}(x)(j)$, and continuing,

$$E^{sim}_{V'} \leq \Big(E^{real}_{V'} + \mu'(|x|)\Big) \sum_{j}^{\infty} \Pr\Big[\#q_\bullet \geq j\Big] =$$
$$\Big(E^{real}_{V'} + \mu'(|x|)\Big)\mathbf{Exp}[\#q_\bullet] =$$
$$\Big(E^{real}_{V'} + \mu'(|x|)\Big)p_2(|x|) =$$
$$p_2(|x|)E^{real}_{V'} + \mu''(|x|)$$

where $\mu''(n)$ is a negligible function. We conclude that the total combined expected running-time of $S$ and $V'$ is

$$p_1(|x|) + p_2(|x|)E^{real}_{V'} + \mu''(|x|)$$

■

REMARK: Using an argument of [36], we note that the lemma cannot be strengthened to consider *all* black-box simulators: Simply consider a verifier $V'$ that always performs a small amount of computation, except upon receiving a particular message $m$ (which the honest prover only sends with exponentially small probability); upon receiving this special message it instead performs a "very large" amount of computation. Now, consider a simulator $S$ that *always* feeds its oracle the query $m$. The running-time of the simulator is thus always large, whereas the expected time of the verifier is small.

By observing that the simulators of [29, 25, 28, 8] are natural, we directly get:

*Theorem* 7. Assume the existence of one-way functions. Then, there exists a polynomial $p(n,t)$ such that:

1. Every language in $\mathcal{NP}$ has an (efficient-prover) computational $\mathcal{ZK}$ proof with expected precision $p(n,t)$.

2. Every language in $\mathcal{NP}$ has an (efficient-prover) constant-round computational $\mathcal{ZK}$ argument with expected precision $p(n,t)$.

3. Every language in $\mathcal{IP}$ has a computational $\mathcal{ZK}$ interactive proof with expected precision $p(n,t)$.

*Theorem* 8. Assume the existence of constant-round statistically (perfectly) hiding commitments. Then, there exists a polynomial $p(n,t)$ such that:

1. Every language in $\mathcal{NP}$ has an (efficient-prover) constant-round computational $\mathcal{ZK}$ proof with expected precision $p(n,t)$.

2. Every language in $\mathcal{NP}$ has an (efficient-prover) constant-round statistical (perfect) $\mathcal{ZK}$ proof with expected precision $p(n,t)$.

EXISTENCE OF $\mathcal{ZK}$ WITH HIGHER-ORDER MOMENT PRECISION. We show that natural *strict* polynomial-time black-box simulators (i.e., simulators whose running-time is polynomially-bounded in the worst-case, and not in expectation) guarantee $\mathcal{ZK}$ also with $m$'th moment precision for $m \in N$. By natural we here mean a strict polynomial-time simulator having the following properties: 1) there exists a round-function $\mathtt{round}_n(\cdot)$ such that on input a statement $x \in \{0,1\}^n$, the $j$'th query of the simulator is always a partial transcript containing $i = \mathtt{round}_n(j)$ rounds, and 2) each such query is indistinguishable from an $i$-round transcript of an interaction between the honest prover and the verifier. More precisely,

*Definition* 16 *(Natural Strict Black-box Simulators).* Let $(P,V)$ be a $k$-round black-box $\mathcal{ZK}$ proof (argument) system for the language $L \in \mathcal{NP}$ with the witness relation $R_L$, and let $S$ denote its black-box simulator. We say that $S$ is a *natural strict black-box simulator,* if $S$ is a probabilistic polynomial-time machine, and there exists a function $\mathtt{round}_n : [n] \to [k(n)]$, such that for every probabilistic polynomial-time interactive machine $V'$, the following two ensembles, indexed by $x \in L, y \in R_L(x), z, r \in \{0,1\}^*, j \in N$, are computationally indistinguishable over $L$.

- $\left\{ view \leftarrow \mathrm{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \ : \ view^{\mathtt{round}_{|x|}(j)} \right\}$

- $\left\{ view \leftarrow \mathtt{query}_{S_\bullet^{V'_r(x,z)}(x)}(j) \ : \ view \right\}$

where $\mathtt{query}_{S_{r'}^{V_r'(x,z)}(x)}(j)$ denotes the $j$'th (partial) view feed by $S_{r'}$ to $V_r'(x,z)$ (or $\perp$ if $S_{r'}$ feeds less than $j$ queries to $V_r'(x,z)$) and $view^i$ denotes the partial view consisting of the first $i$ rounds in the view $view$.

*Lemma* 2. Let $S$ be a natural strict black-box simulator for the perfect/statistical/ computational $\mathcal{ZK}$ proof systems $(P,V)$ for $L$ with witness relation $R_L$. If $S$ has running-time $p_1(n)$ and makes $p_2(n)$ queries to its oracle, then there exists a negligible function $\mu(n)$ such that $S$ has $m$'th-moment precision

$$p(n,t) = \Big(p_2(n)+1\Big)^m t + \Big(p_2(n)+1\Big)^m p_1(n)^m + \mu(n)$$

**Proof:** Consider any probabilistic polynomial-time adversary verifier $V'$. Fix generic $x \in L, w \in R_L(x), z, r \in \{0,1\}^*$. Let $M_{V'}^{(m)}$ denote the $m$-order moment of the running time distribution of $V_r'(x,z)$ in an interaction with $P_\bullet(x,w)$. Let $M_S^{(m)}$ denote the $m$'th-order moment of the running time distribution of $S_\bullet(x,z)$. As in the proof of lemma 1, to simplify notation, we let

$$q_\bullet(j) = \mathtt{query}_{S_\bullet^{V_r'(x,z)}(x)}(j)$$

By definition,

$$M_S^{(m)} = \mathbf{Exp}\Big[(\text{STEPS}_{S_\bullet^{V_r'(x,z)}})^m\Big] \leq$$

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j)\ :\ \Big(p_1(|x|) + \sum_{j}^{p_2(|x|)} \text{STEPS}_{V'}(v_j)\Big)^m\Big]$$

(We emphasize that in the above expression the queries $v_1, v_2, \ldots$ are generated from the *same* execution of $S$ with a *particular* random tape. In other words, $q_\bullet(1), q_\bullet(2), ..$ refer to the same execution of $S_\bullet^{V_r'(x,z)}(x)$.)

By applying Hölders inequality[3] and by the monotonicity of expectations, we get

$$M_S^{(m)} \leq$$

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j)\ :\ \Big(p_2(|x|)+1\Big)^{m-1}\Big(p_1(|x|)^m + \sum_{j}^{p_2(|x|)} \text{STEPS}_{V'}(v_j)^m\Big)\Big] =$$

$$\Big(p_2(|x|)+1\Big)^{m-1}\Big(p_1(|x|)^m + \mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j)\ :\ \sum_{j}^{p_2(|x|)} \text{STEPS}_{V'}(v_j)^m\Big]\Big)$$

By linearity of expectation, we get

$$M_S^{(m)} \leq$$

$$\Big(p_2(|x|)+1\Big)^{m-1}\Big(p_1(|x|)^m + \sum_{j}^{p_2(|x|)} \mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j)\ :\ \text{STEPS}_{V'}(v_j)^m\Big]\Big) \quad (4.2)$$

[3]Recall that Hölder's inequality states that $\sum |x_i y_i| \leq (\sum |x_i|^p)^{1/p}(\sum |y_i|^q)^{1/q}$ when $1/p + 1/q = 1$, which in particular means that $(\sum_i |x_i|)^m \leq (\sum_i |x_i|^m)(\sum_i 1)^{m-1}$.

We next show, in the same way as in the proof of lemma 1 that there exists a negligible function $\mu(n)$ such that

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m\Big] \le M_{V'}^{(m)} + \mu(|x|)$$

As in lemma 1,

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m\Big] = \sum_t^{\infty} \Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m \ge t\Big] = \sum_t^{\text{TIME}_{V'}(|x|)^m} \Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m \ge t\Big]$$

It follows, due to the natural simulation property of $S$, and due to the fact that $\text{STEPS}_{V'}$ is efficiently computable (since it is bounded by $\text{TIME}_{V'}(n)$, which in turn is bounded by a polynomial), that there exists a negligible function $\mu'(n)$ such that for all $t \le \text{TIME}_{V'}(|x|)^m$

$$\Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m \ge t\Big] \le \Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z)] : \text{STEPS}_{V'}(v^{\text{round}_{|x|}(j)})^m \ge t\Big] + \mu'(|x|) \le \Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z)] \ : \ \text{STEPS}_{V'}(v)^m \ge t\Big] + \mu'(|x|)$$

Since $\text{TIME}_{V'}(n)^m$ is a polynomial, we conclude that there exists a negligible function $\mu(n)$ such that

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \text{STEPS}_{V'}(v_j)^m\Big] \le \mu(|x|) + \sum_t^{\text{TIME}_{V'}(|x|)^m} \Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z)] : \text{STEPS}_{V'}(v)^m \ge t\Big] = \mu(|x|) + \mathbf{Exp}\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z)] \ : \ \text{STEPS}_{V'}(v)^m\Big] \le M_{V'}^{(m)} + \mu(|x|)$$

Plugging this into equation 4.2 and continuing, we get

$$M_S^{(m)} \le \Big(p_2(|x|)+1\Big)^{m-1} \cdot \Big(p_1(|x|)^m + \sum_j^{p_2(|x|)} (M_{V'}^{(m)} + \mu(|x|)\Big)$$
$$\le \Big(p_2(|x|)+1\Big)^m M_{V'}^{(m)} + \Big(p_2(|x|)+1\Big)^m p_1(|x|)^m + \mu''(|x|)$$

for some negligible function $\mu''$. ■

By observing that the strict polynomial-time simulators of the protocols of [29, 11, 8] are natural, we directly get:

*Theorem* 9. Assume the existence of one-way functions. Then, then for every $i \in N$, there exists a polynomial $p(n,t)$ such that:

1. Every language in $\mathcal{NP}$ has an (efficient-prover) computational $\mathcal{ZK}$ proof with $i$'th moment precision $p(n,t)$.

2. Every language in $\mathcal{IP}$ has a computational $\mathcal{ZK}$ interactive proof with $i$'th moment precision $p(n,t)$.

*Theorem* 10. Assume the existence of statistically (perfectly) hiding commitments. Then, then for every $i \in N$, there exists a polynomial $p(n,t)$ such that every language in $\mathcal{NP}$ has an (efficient-prover) statistical (perfect) $\mathcal{ZK}$ proof with $i$'th moment precision $p(n,t)$.

We mention that whereas in the case of $\mathcal{ZK}$ with expected precision we are able to obtain *constant-round* protocols, we do not know whether constant-round protocols that are $\mathcal{ZK}$ with higher-moment precision can be constructed. We note that it is unlikely that lemma 2 can be useful in the construction of such protocols, as Barak and Lindell [6] show that only languages in $\mathcal{BPP}$ have constant-round $\mathcal{ZK}$ arguments with *strict* polynomial-time black-box simulators.

We furthermore mention that the simulators for the constant-round protocols of Feige and Shamir [25] and Goldreich and Kahan [28] do not have polynomial second-moment precision: These simulators proceed by first running the malicious verifier up until a certain round in the protocol. If the verifier has not aborted in this first execution, they rewind the verifier and feed it new messages until they get another execution where the verifier does not abort. Consider a verifier that aborts with probability $1-p$. With probability $p$ such simulators obtain in their first execution a view in which the verifier does not abort. We conclude that the second moment of their running-time is *at least*

$$p\sum_{i=1}^{\infty} i^2(1-p)^{i-1}p = p\frac{O(1)}{p^2} = \frac{O(1)}{p}$$

Since the simulators should work for all verifiers of the above type, where $p$ is an arbitrary inverse polynomial, we conclude that the second moment of the running-time of the simulators is not bounded by any fixed polynomial.

5

# Capturing Knowledge Precisely

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

## 5.1 Precise Zero Knowledge

*Definition* 17 *(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:

   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 $r \in \{0,1\}^*$, $\text{STEPS}_{S_r(x,z)} \le p(|x|, \text{STEPS}_{V'}(S_r(x,z))$.

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).[1]

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.[2] 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 $\Omega(n^{50})$; we conclude that even a simulator with optimal expected precision could potentially *always* take $\Omega(n^{50})$ steps, whereas $V'$ almost always takes $n$ steps. In fact, even a simulator with optimal higher-moment precision might always take $\Omega(n^{50})$ 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.

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

[1]This is not a new observation. Indeed, the definition of statistical *black-box* zero-knowledge of [58] calls for simulation of also unbounded verifiers.

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

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.

## 5.2 Properties of Precise ZK

### 5.2.1 Preserving Running-time Distribution

Whereas the notion of knowledge-tight $\mathcal{ZK}$ (see Definition 3.1.1) 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. Note that this is stronger than our requirement of expected precise $\mathcal{ZK}$, which only guarantees that the simulator's expected running-time respects the expected running-time of the verifier. (In fact, as we show below, precise $\mathcal{ZK}$ also implies $i$'th moment ($i \in N$) precision.)

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 $t \in N$:

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

where

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

and

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

**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:

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

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 $\text{STEPS}_{V'}(v) \le \text{TIME}_{V'}(|x|)$ and $\text{STEPS}_{V'}(v) \le t$.

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

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

and

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

PRESERVING HIGHER-ORDER MOMENTS. In the sequel of the thesis we will mainly be interested in precision functions of the form $p(n,t) = p'(n)t$ (in particular all our construction have such precision). As a corollary to Proposition 2 we directly get that for all protocols that are $\mathcal{ZK}$ with precision $p(n,t) = p'(n)t$, it holds that for all probabilistic *polynomial-time* verifiers $V'$, the $i$'th moment of the running-time distribution of the simulator for $V'$ is at most $p_2(n)^i$ times the $i$'th moment of the running-time distribution of $V'$ in a real interaction with the prover.

*Corollary* 1. Let $(P,V)$ be a perfect/computational/statistical $\mathcal{ZK}$ proof (or argument) for the language $L$ with witness relation $R_L$, and precision $p(n,t) = p'(n)t$, where $p'(n)$ is a monotonically increasing function. Then there exists a negligible function $\mu$ such that $(P,V)$ is $\mathcal{ZK}$ with $i$'th moment precision $p(n) = (p'(n))^i t + \mu(n)$.

**Proof:** Consider any probabilistic polynomial-time ITM $V'$ and its corresponding simulator $S$. For any $x \in L$, $z \in \{0,1\}^*$, let $M_{V'}^{(i)}(x,z)$ denote the $i$'th moment of the running-time distribution of $V'_\bullet(x,z)$ in an interaction with $P_\bullet(x,y)$ for any $y \in R_L(x)$, and let $M_S^{(i)}(x,z)$ denote the $i$'th moment of the running-time distribution of $S_\bullet(x,z)$. We start by noting that

$$M^{(i)}(x,z) =$$

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

$$\sum_t^{\text{TIME}_{V'}(|x|)^i} \Pr\left[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z) \ : \ (\text{STEPS}_{V'}(v))^i \geq t\right] =$$

$$\sum_t^{\text{TIME}_{V'}(|x|)^i} \Pr\left[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V_r'(x,z) \ : \ (\text{STEPS}_{V'}(v)) \geq t^{1/i}\right]$$

By applying Proposition 2 we get that there exists some negligible function $\mu'(n)$ such that

$$M^{(i)}(x,z) \leq \sum_t^{\text{TIME}_{V'}(|x|)^i} \left(\Pr\left[\frac{\text{STEPS}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\right] + \mu(|x|)\right) =$$

$$\mu(|x|) \cdot \text{TIME}_{V'}(|x|)^i + \sum_t^{\text{TIME}_{V'}(|x|)^i} \left(\Pr\left[\frac{\text{STEPS}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\right]\right)$$

Since $V'$ is a polynomial-time machine, there thus exists a negligible function $\mu'(n)$ such that

$$M^{(i)}(x,z) \leq \mu'(|x|) + \sum_t^{\text{TIME}_{V'}(|x|)^i} \Pr\left[\frac{\text{STEPS}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\right]$$

$$= \mu'(|x|) + \sum_t^{\text{TIME}_{V'}(|x|)^i} \Pr\left[\frac{(\text{STEPS}_{S_\bullet(x,z)})^i}{p'^i(|x|)} \geq t\right] =$$

$$\mu'(|x|) + \frac{1}{p'^i(|x|)} \sum_t^{\text{TIME}_{V'}(|x|)^i} \Pr\left[S_\bullet(x,z)^i \geq t\right] =$$

$$\frac{M_S^{(i)}(x,z)}{p'^i(|x|)} + \mu'(|x|)$$

■

### 5.2.2 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 [30] have shown that the standard definition of $\mathcal{ZK}$ (see Definition 8) 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 [30] can by used to show that the protocol resulting from sequentially repeating a precise $\mathcal{ZK}$ also is precise a $\mathcal{ZK}$ proof (albeit with slightly worse precision).

*Lemma* 3 *(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 [28] (see also Feige and Shamir [24]) 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 Chapter 6) suffices to show that also the notion of precise $\mathcal{ZK}$ is not closed under parallel repetition.

## 5.3 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* 18 *(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 18 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.

## 5.4 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 in Section 5.4.1 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 alleged 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* [42], it differs in several crucial ways, as will be discussed shortly.

*Definition* 19 *(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 19 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 [42] 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.

### 5.4.1 $\mathcal{ZK}$ 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* 20 *(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* 4. 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 $O(1) \cdot [p_1(n, t) + p_2(n, t)]$. ■

6

# Existence of Precise ZK

In this chapter we present our main results regarding the existence of precise $\mathcal{ZK}$ proof and arguments.

It should be appreciated that without any trusted set-up assumptions, none of the known zero-knowledge protocols and simulators satisfy our precise simulation requirement. In particular, in Section 6.5, 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 A.1.[1]

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 statistically-hiding commitments. Then, every language in $\mathcal{NP}$ has:*

*1. an $\omega(k(n))$-round computational $\mathcal{ZK}$ proof with polynomial precision.*

[1]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. In fact, such a non-black box simulator is not even sufficient to meet our notion of expected precise $\mathcal{ZK}$.

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 $\mathcal{NP}$.*

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, $\mathbf{SD}^1_{1/2}$ [55, 46]. (The general, i.e., non-restricted, Statistical Difference problem is complete for statistical $\mathcal{ZK}$ [55])

OUR NEW SIMULATION TECHNIQUE. To prove our positive results, while avoiding our impossibility ones, we rely on a simulation that *counts the number of steps* taken by a verifier, and *times out* the verifier when it runs for too long. Notice that both these operations entail a non-black box simulation, even if in a minimal sense.

On a very high level (and grotesquely oversimplified) the idea behind our simulation technique is to let the simulator *learn* (in a very weak sense) the running-time distribution of the verifier, and then *exploit* it to perform a precise simulation. The main obstacle that arises is that the "learning"-phase must be done very efficiently, i.e., the simulator cannot take (much) more time than the verifier it is sampling.

OVERVIEW. In Section 6.1 we provide an overview of our construction paradigm. Sections 6.2, 6.3, 6.4 are devoted to the actual constructions. In Section 6.5 we demonstrate our black-box lower bounds.

## 6.1 A Common Paradigm

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 ($\mathcal{WI}$) [23] 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* [15].[2] (Different lemmas state that different number of repetitions yield different levels of precision—the higher the number the better the precision.) Following the original work of Richardson and Kilian [53], (see also [40, 52]), repetition of a special-sound proof of knowledge provides the extractor the ability to compute a valid witness in polynomial time, even in a presence of multiple *concurrent executions*. In our application, we do not have to worry about concurrency, but must instead provide a much tighter running time for the extractor: "polynomial time" is not good enough.

On a high-level our extractor proceeds as follows on input a view of the prover $P'$. Initially, the extractor feeds the view to $P'$ and lets $P'$ run undisturbed (letting him take as much time as he pleases) in each of the $m$ sub-protocols. While doing this, the extractor keeps track of the number of computational steps $P'$ takes in each of the sub protocols. Soon after, the extractor will attempt to compute a witness by rewinding each of the $m$ sub-protocols a *fixed* (i.e., independent of the prover $P'$ and in particular its running time) number of times. Moreover, our extractor aborts each such rewinding as soon as $P'$'s running time exceeds its running time in the first execution of *that* very sub-protocol. When we succeed in extracting a witness in this fashion, we automatically guarantee that the running time of the extractor is close to that of $P'$ in its first execution, and thus close to the running-time $P'$ on the view received as input. The added difficulty of our approach, however, consists in proving that such constrained extraction will essentially always succeed.

## 6.2 Knowledge Precision Lemmas

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

[2]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.

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 [23]).

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 $(P, V)$.

### 6.2.1 Statistical Knowledge Precision Lemmas

We start by recalling the notion of *special-sound* proofs [15]. (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 $\text{ACCEPT}_V$ 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* 21 *(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 $\text{ACCEPT}_V$ can be computed in linear time (in its inputs length) and the extractor $X$ in definition 21 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 [15, 16]. We here show that appropriate sequential repetition of a special-sound proof results in an precise proof of knowledge.

**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* 5 *(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 step 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.[3]

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 $r'_i \overset{R}{\leftarrow} \{0,1\}^l$, 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 $w$.

[3]As far as we know this is the first non black-box reduction that uses the code of the adversary to actually "learn" something about the adversary. Previous non-black box reductions (c.f. [2]) only use the code of the adversary in order to "diagonalize" against it.

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 [23] that $(P', V')$ is $\mathcal{WI}$ if $(P, V)$ is so. ■

**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* 6 *(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 5, 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 5 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 5: Given any view $view_{P'}$, let $view_{P'}^i$ 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 $steps(view_{P'}^i, a)$ denote the number of steps $P'$ takes to correctly answer the $i$'th challenge, when first feed the view $view_{P'}^i$, and then the $i$'th challenge $a$.

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 $r_i^{(j)}$ the $j$'th new challenge sent by $E$):

1. $r_i = r_i^{(j)}$ for all $j \in [n]$

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 $\mu$ 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:

- $\Big(a, (a^{(1)}, .., a^{(n)})\Big)$ and $\Big(b, (b^{(1)}, .., b^{(n)})\Big)$ 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 $\Big(a, (a^{(1)}, .., a^{(n)})\Big)$ and $\Big(b, (b^{(1)}, .., b^{(n)})\Big)$ 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[4] $\Big(a, (a^{(1)}, .., a^{(n)})\Big)$ and every prefix view $v$ such 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 $\Big(a^{(1)}, (a, a^{(2)}, .., a^{(n)})\Big)$, $\Big(a^{(2)}, (a^{(1)}, a, .., a^{(n)})\Big)$,...., $\Big(a^{(n)}, (a^{(1)}, .., a^{(n-1)}, a)\Big)$. (Note that all the above classes indeed are distinct since we only consider 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

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

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

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

As in the proof of lemma 5, we conclude that the probability (over a random input view $view_{P'}$ and the internal random coins of $E$) that extraction fails in $i$'th execution

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

Again, as in the proof of lemma 5, 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 5 it directly follows that $(P', V')$ is $\mathcal{WI}$ if $(P, V)$ is so. ■

### 6.2.2 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 statistically hiding commitments is known to be computationally special-sound.)

We start by formally defining computational special-soundness. Recall the definitions of ACCEPT, consistency from Section 6.2.1. Furthermore, let CONSISTENT denote the predicate that on input two transcripts outputs 1 if and only if the transcripts are consistent.

*Definition* 22 *(Computationally Special-sound Proofs).* 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 *computationally 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 $X$, and a negligible function $\mu$, such that for all probabilistic polynomial-time machine $G$

$$\Pr\left[(x,T_1,T_2) \leftarrow G(x) \;:\; \begin{array}{c} \text{ACCEPT}_V(x,T_1) = 1 \\ \text{ACCEPT}_V(x,T_2) = 1 \\ \text{CONSISTENT}(T_1,T_2) = 1 \\ X(x,T_1,T_2) \notin R_L(x) \end{array}\right] \leq \mu(|x|)$$

If, furthermore, the predicate $\text{ACCEPT}_V$ 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* 7 *(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* 8 *(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 7 and Lemma 8:** Both lemmas follow essentially directly from the proofs of Lemma 6 and Lemma 5. 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 6 and Lemma 5 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 6 and Lemma 5 is negligible. By the union-bound we thus get that the total failure probability also is negligible. This concludes the proof of Lemma 7 and Lemma 8. ■

## 6.3 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.

### 6.3.1 $\mathcal{WI}$ Precise Proofs of Knowledge

By combining Lemma 6 and Lemma 5 with the Blum's proof system for Hamiltonicity [9] (relying on [38] and [48]), we obtain:

*Theorem* 11. 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 [9] is a special-sound proof (in fact it was the first special-sound proof system known for $\mathcal{NP}$). Since the proof system is $\mathcal{ZK}$, it is also $\mathcal{WI}$ [23]. Furthermore, since $\mathcal{WI}$ is closed under parallel composition [23], 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 6.

LINEAR PRECISION CASE. It can be seen that, if using an appropriate representation of 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 5. ■

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

By instead combining Lemma 7 and 8 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* 12. Assume the existence of $k(n)$-round 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 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 11. ■

### 6.3.3 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 11 and 12 by applying Lemma 4.

*Theorem* 13. 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* 14. Assume the existence of $k(n)$-round 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.

## 6.4 Constructions of Precise 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.

### 6.4.1 Precise $\mathcal{ZK}$ Arguments for $\mathcal{NP}$

We start by showing the following theorem :

*Theorem* 15. Assume the existence of $k(n)$-round 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 reject 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 $f : \{0,1\}^n \rightarrow \{0,1\}^n$ 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 depicted in Figure 6.1. Note that the protocol relies on the existence of one-way functions, 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 13, 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 $k(n) + \omega(1)$.

*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 [24] as protocol *StatPolyPreciseZKArg* is a particular

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

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 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 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[5] 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

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

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).

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 6.2. Since by Theorem 13 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 $w'$.

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

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

V $\to$ 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 6.2: Statistical $\mathcal{ZK}$ argument for $\mathcal{NP}$ with linear precision

*Proposition* 4. Protocol *LinPreciseStatZKArg* 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 15. ■

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

Just as in the protocol of Feige and Shamir [24], 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 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 [23, 29, 48, 38], we thus obtain:

*Theorem* 16. 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 15. If relying on the same simulator $S$ as in the proof of Theorem 15, 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 $L$

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

■

### 6.4.2 Precise $\mathcal{ZK}$ Proofs for $\mathcal{NP}$

We show:

*Theorem* 17. 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 [28]. 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 [52, 54].) 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 6.3.

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 and since by Theorem 14, the existence of a $k(n)$-round 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 $\omega(k(n))$.

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: $c = \textsc{Com}(\overline{r}; s)$, where $\textsc{Com}$ 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 $c = \textsc{Com}(\overline{r}'; s')$

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

**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 6.3: Computational $\mathcal{ZK}$ Proof for $\mathcal{NP}$ with Polynomial Precision

*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 [28]. 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.[6] Thus, Soundness and Completeness

[6]If this was not the case there would exists at least one transcript of the commitment $\textsc{Com}$, such that the $\mathcal{WI}$ proof of knowledge protocol reveals (to an unbounded receiver) what value the transcript is a commitment to. This, however, contradicts the statistical-$\mathcal{WI}$ property of the proof of knowledge protocol.

of *CompPolyPreciseZKProof* follows directly from the proof of Goldreich and Kahan [28]. 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:

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 [28], $S$ 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 $\mathcal{ZK}$.)[7]
   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 $L$.

[7]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.

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

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 $L$

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

**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:** To prove claim 5, consider an additional hybrid simulator $S''$ that on input $x$, $(y,z)$ proceeds as follows:

1. $S''$ first runs $S_\bullet(x,z)$.
2. If $S$ outputs `fail`, $S''$ halts outputting `fail`.
3. Otherwise, it runs $S'_\bullet(x,(y,z))$ and outputs whatever $S'$ outputs.

We start by noting that it follows directly from the computational-binding property of the commitment scheme used in Stage 1, that the output of $S'$ is statistically close to the output of $S''$. 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, which implies that $S_\bullet(x,z)$ and $S'_\bullet(x,(y,z))$ also 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 *CompPoly-PreciseZKProof* 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 14 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 *CompLin-PreciseZKProof* is computational $\mathcal{ZK}$ with linear precision. This concludes the proof of Theorem 17. ∎

### 6.4.3 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* 18. 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 [8] 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); see Section 3.2.2 for more details. 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 [8], 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).[8] (The prover directly aborts the proof if $V$ sends a string that is shorter.) ■

### 6.4.4 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 [29] 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.

#### 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 [29] and the protocol of Benaloh [3]) is depicted in Figure 6.4.

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

**Proof:** Soundness and Completeness follow directly using the same proof as in [29]. Statistical-$\mathcal{WI}$ follows from the fact that protocol *1of2GraphIsoProof* is honest-verifier perfect zero-knowledge (see [29]), 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 $\Omega(n)$.

[8]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 of $C_i, C_{1-i}$.

V checks the validity of the permutations received.

Figure 6.4: Statistically Precise $\mathcal{ZK}$ proof for 1OF2GRAPHISO

By combining Proposition 7 with Lemma 6 and Lemma 5 we obtain:

*Theorem* 19. 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 19 by applying Lemma 4.

*Theorem* 20. 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.

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

Let GRAPHNONISO denote the language of non-isomorphic graphs.

*Theorem* 21. 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 6.4.4. Consider the protocol depicted in Figure 6.5 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 6.5: 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 [29]. $\mathcal{ZK}$ 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 15. ■

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

The same approach as above can directly be applied to Goldwasser, Micali and Rackoff's [33] protocol for Quadratic Non-Residuosity, QNR. Furthermore, by instead relying on a protocol of Micciancio, Ong, Sahai, and Vadhan [45] (extending [52] and [46]) we can obtain statistical precise $\mathcal{ZK}$ proof with polynomial and linear precision for all problems in $\mathbf{SD}^1_{1/2}$ [46, 55]. (The general, i.e., non-restricted, Statistical Difference problem is complete for statistical $\mathcal{ZK}$ [55]). We note that although GRAPHNONISO and QNR actually reduces to $\mathbf{SD}^1_{1/2}$, the protocol resulting from relying on the protocol of [45] require a "large" round-complexity and non-efficient provers, whereas our direct approach avoids this.

## 6.5 Black-Box Lower Bounds

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.

### 6.5.1 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 15 in Section 3.1.1). 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* 23 *(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.)

### 6.5.2 The Lower Bound

We prove the following theorem:

*Theorem* 22. 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,

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

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

1. First, note that Theorem 22 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 22 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 22 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 $\mathcal{NP}$ [51] (which are based on the existence of one-way functions with sub-exponential hardness) are black-box zero-knowledge with *quasi-polynomial* precision.[9]

**Proof of Theorem 22:** 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 $\left(p(n, T(n))\right)^2$ 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

[9]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.

$\big(p(n, T(n))\big)^2$ 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 $\text{TIME}_V$ 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.

*Claim* 6. Unless $L \in \mathbf{BPTIME}[O(p(n, \text{TIME}_V(n)))]$, 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 $\mu$ 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 $D_{\bullet}(x) \rightarrow 1$ 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. We construct a cheating prover $P'$ as follows.

1. $P'$ internally runs $S_{\bullet}(x)$.
2. Whenever $S$ makes a query to its oracle, $P'$ checks that $S$ has not previously asked the same query (or a subset of it). If it has not (i.e., if it is a new query) $P'$ externally forwards the prover messages in the query.

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 $V'$. 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 22.

*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. ■

■

EXTENSION 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 $\mathcal{NP}$.

A I

# Appendix

## A.1 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 [43]. 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 $T : N \rightarrow N$ be a "nice" function that satisfies $T(k) = k^{\omega(1)}$. A high-level overview of Barak's protocol is depicted in Figure A.1.

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

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 $h \overset{R}{\leftarrow} \mathcal{H}_k$.

**Stage 1:**

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

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

Figure A.1: Barak's Non Black-Box ZK Argument for $\mathcal{NP}$

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 – as such this simulator even has "bad" *expected* precision).

# Bibliography

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

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

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

[18] 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. G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge

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

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

[21] 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.

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

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

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

[25] 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.

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

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

[28] 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.

[29] 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.

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

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

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

[33] 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.

[34] 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.

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

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

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

[38] 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.

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

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

[41] 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.

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

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

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

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

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

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

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

[49] 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.

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

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

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

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

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

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

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

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

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