Precise Cryptography

Silvio Micali
CSAIL, MIT
silvio@csail.mit.edu

Rafael Pass
Cornell University
rafael@cs.cornell.edu

September 4, 2007

Abstract

Precise zero knowledge guarantees that the view of any verifier V can be simulated in time closely related to the *actual* (as opposed to worst-case) time spent by V in the generated view. We generalize this notion in two ways:

1. We provide definitions and constructions of *precise encryption*, *precise proofs of knowledge* and *precise secure computation*.
2. We introduce relaxed notions of precise zero knowledge—which relate the expectation (or higher moments) of the running time of the simulation with the expectation (higher-moments) of the running-time of the verifier—and provide conditions under which these notions are implied by the traditional (non-precise) notion of zero knowledge.

1 Introduction

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali and Rackoff [18] are 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 no additional knowledge to the Verifier. The zero-knowledge property is formalized by requiring that the view of any PPT verifier V in an interaction with the prover—can be "indistinguishably reconstructed" by a PPT simulator S—interacting with no one—on input just x. The rational behind this definition is that since whatever V "sees" in the interaction can be reconstructed by the simulator the interaction does not yield anything to V that cannot already be computed by simply running the simulator. Since the simulator is allowed to be an arbitrary PPT machine, this traditional notion of $\mathcal{ZK}$ thus only guarantees that the *class* of PPT verifiers learn nothing. To measure the knowledge gained by a particular verifier, Goldreich, Micali and Wigderson [15] (see also [12]), put forward the notion of *knowledge tightness*: intuitively, the "tightness" of a simulation is a function relating the (worst-case) running-time of the verifier and the (expected) running-time of the simulator—thus, in a knowledge-tight $\mathcal{ZK}$ proof, the verifier is guaranteed not to gain more knowledge than what it could have computed in time closely related to its *worst-case* running-time.

Pass and Micali [26] recently introduced the notion of *precise zero knowledge* (originally called *local* $\mathcal{ZK}$ in [26]). In contrast to traditional $\mathcal{ZK}$ (and also knowledge-tight $\mathcal{ZK}$), precise $\mathcal{ZK}$ considers the knowledge of an individual verifier in an *individual execution*—it requires that the view of any verifier V, in which V takes t computational steps, can be reconstructed in time closely related to t—say $2t$ steps. This notion thus guarantees that the verifier does not learn more than what can be computed in time closely related to the *actual* time it spent in an interaction with the prover. As such, it is applicable also when e.g., considering knowledge of "semi-easy" properties of the instance x, or even when considering proofs for "semi-easy" languages L (see [29] for more details).

In this paper, we generalize this notion in two ways:

1. We consider notions of precision also for primitives such as encryption, proofs of knowledge and secure computation.

2. We consider relaxed notions of precision and investigate under what circumstances we these notions are implied by traditional security notions.

1.1 Precision Beyond $\mathcal{ZK}$

Precise Encryption We provide a strengthening of the traditional semantical security definition of [17]. In fact, our strengthening applies also to Shannon's original definition of prefect secrecy—as far as we know, this is the first strengthening of Shannon's definition (when considering single-message security under a passive attack). Intuitively, (for the case of public-key encryption)

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

Recall that the definition of [17] only requires that the eavesdropper learns no more than what could have been computed in polynomial time. Due to the equivalence between semantical-security and indistinguishability [17], it is, however, easy to see that any semantically-secure encryption scheme has precision $p(n,t) = O(t + g(n))$ where g is a polynomial upper-bounding the running-time of the

encryption algorithm.[1] Thus, semantically-secure encryption schemes are already "quite" precise (in contrast to traditional $\mathcal{ZK}$ [26]).[2]

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

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

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

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

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

Precise Proofs of Knowledge The notion of a proof of knowledge was intuitively introduced by Goldwasser, Micali and Rackoff [18] and was formalized by Feige, Fiat and Shamir [9] and Tompa and Woll [31]. Loosely speaking, an interactive proof of $x \in L$ is a proof of knowledge if the prover can only convinces the verifier (with non-negligible probability), if it in PPT can compute a witness w for the statement x. This is formalized by requiring the existence of a PPT machine E, called an *extractor*, such that E on input the description of P and any statement $x \in L$ outputs a valid witness for $x \in L$ if P succeeds in convincing the Verifier (with non-negligible probability) that $x \in L$. Halevi and Micali [21] 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. Their notion thus guarantees that P will only be able to convince the Verifier that $x \in L$, if it could have computed a witness for $x \in L$ in time closely related to its expected running-time.

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

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

[2] However, this statement (although potentially obvious to experts) can only be formalized once we have a definition of precise encryption.

Towards this goal, we put forward a notion of a proof of knowledge that more precisely bounds the knowledge of the prover in an *execution by execution* manner: Consider for instance a teacher wishing to verify whether a student has done its homework. We here require the use of a testing procedure that checks if the student indeed did its homework (and knows what it is supposed to know), and not merely that the student *could* have done it (under some different circumstances).[3] Thus we would like to have a notion of a proof of knowledge which requires that a prover (the student) can only convince the verifier (the teacher) whenever it "essentially" has a witness on its worktape (its brain) at the end of the interaction (the exam). More precisely, we put forward a notion of a proof of knowledge which guarantees that

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

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

We mention that in [26] we already introduced a different notion of a proof of knowledge; this notion—which is called a *precisely emulatable* proofs of knowledge in [29], and whose definition is similar in spirit to *witness-extended emulation* [24]— did not have an as clear semantics and was merely used as a technical tool in the construction of precise $\mathcal{ZK}$ proofs (See Section 3 and [29] for more details). Nonetheless, as we show here, the same protocols as in [26] can be showed to satisfy our notion of a precise proof of knowledge.

Precise Secure Computation We finally provide a definition of precise secure computation. Whereas the definitions of precise encryption and precise proofs of knowledge are quite different from the traditional definitions, the definition of precision secure computation is instead "just" a careful adaptation of the precise $\mathcal{ZK}$ definition to the setting of secure computation. Additionally, we show that by appropriately compiling the GMW protocol [15] using precise $\mathcal{ZK}$ protocols (and applying careful padding), results in a secure computation protocol with linear precision. Similar modifications can be done also to the protocols of [2] and [30].

1.2 Expected and Higher-Moment Precision

As is shown in [29], the notion of precise $\mathcal{ZK}$ guarantees that the *distribution* of the running-time of the simulator "respects" the distribution of the running-time of the verifier (and not just is worst-case running-time). A natural question that arises is whether it is easier to satisfy a notion of precision that only guarantees that the distribution of the simulator only respects certain statistical properties—such the expectation, or higher-moments—of the running-time distribution of the verifier.

We can view the expected running-time of a verifier as its expected "cost" of computation; at the very least we would expect that a $\mathcal{ZK}$ proof guarantees that the expected cost of some computation for the verifier has not (significantly) decreased after hearing the proof. We also remark that in this

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

context, higher-moments of the running-time distribution of a verifier have natural interpretations.[4] Although one could argue that the actual "objective" cost of a computation (i.e., the cost of the hardware needed to perform some computation) in general is linear in the number of steps of computation, the "subjective" cost of computation for a *particular* agent might conceivably be super-linear. Consider, for instance, using some software for implementing a stock trading strategy; it is very conceivable that e.g., a sorting algorithm that runs k times slower looses k^2 money. Thus, an agent might very well be concerned about higher-moments of its running-time (in the above example, representing the amount of money it looses).

Furthermore, recall that the variance of a random variable X can be written as $\mathbf{Var}[X] = \mathbf{Exp}[X^2] - \mathbf{Exp}[X]^2$. (Thus, a simulator with a polynomial expected precision but super-polynomial second-moment precision has a super-polynomial high variance.) It is conceivable that a *risk-averse* verifier's subjective "cost" of the computation grows with the variance (and thus with the second moments of its running-time) — i.e., a verifier might prefer to always take n^2 steps than taking n steps on the average, but sometimes n^5 steps.

Upper Bounds on Expected Precision In stating our results, we assume familiarity with the notion of black-box simulators (see Appendix B.3 for definitions). Our first results shows that for "natural" black-box simulators it is sufficient to bound the expected running-time of the simulator with respect the *worst-case* running-time of the verifier—essentially the same bound will then also hold w.r.t to the expected running-time of the verifier. 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 [21], showing a corresponding result for proofs of knowledge, if assuming that the distribution of the messages produced by the extractor is *identical* (and not just inextinguishable) to the distribution of true verifier messages. Our result is also closely related to a result by Katz and Lindell [22] showing that every proof system that has a certain strong type of natural simulator (in particular for which indistinguishability holds for super polynomial-time, and with super-polynomially small indistinguishability gap), is also $\mathcal{ZK}$ for *expected* polynomial-time verifiers (as opposed to strict polynomial-time verifiers). Their results can be used to directly show that zero-knowledge proof systems that have such a strong type of simulator, also have a (potentially different) simulator with polynomial expected precision. In contrast, our result applies to *all* known black-box simulator (and also provides tighter bounds).[5] Our techniques are related to both of the above works (and in particular [22]).

Upper Bounds on Higher-moment Precision We next consider higher-moment precision. We first observe that there exist black-box zero-knowledge simulators that do *not* have polynomial *second-moment* precision. In fact, known simulators for *constant-round* zero-knowledge protocols (e.g., [11, 14]) have super-polynomial second-moment precision. We turn to consider more restrictive types of black-box simulators, namely *strict* polynomial-time black-box simulators (i.e., simulators whose running-time is polynomially-bounded in the worst-case, and not in expectation). We show that "natural" strict polynomial-time black-box simulators also have good m'th moment precision for $m \in N$ —roughly speaking, we show that the m'th moment of the running-time distribution of any

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

[5] We mention that the reason we dispense of the super-polynomial indistinguishability property is that we only consider verifier with polynomial running-time (instead of expected polynomial-time as in [22]).

strict polynomial-time black-box simulator—whose running-time is $t(n)$ (not counting the running-time of verifier)—is bounded by $O(p(n)^m t)$ where t denotes the m'th moment of the running-time distribution of the verifier.[6] Again, natural here corresponds to an appropriate notion of specifying that the messages sent by the simulator are indistinguishable from real prover messages; again, all known *strict* polynomial-time black box simulators (such as the simulators for the original non constant-round zero-knowledge protocols, e.g., [15, 5]) satisfy this properly.

As a consequence, our results show that although the more recent zero-knowledge protocols (such as [11, 14]) only require a constant number of rounds (instead of a super constant number of rounds as the original protocols), the "quality" of the simulation is significantly worse when considering verifiers that have non-linear computation cost functions. We leave as an open question whether constant-round protocols that have polynomial higher (or even second) moment precision exists.

Other Applications We mention that the proof of the above results does not rely on the specific notion of a zero-knowledge proof, but if fact applies to *all* black-box simulators that only feed the adversary (which it has oracle access to) messages that are indistinguishable from what the adversary would see in a real interaction. As such, our results directly apply also to secure computations and encryption (and any other cryptographic primitives where simulation is used to define security).

A Note on the Concrete Security of Reductions We note that expected precision might also be important when considering reductions between primitives. Consider for instance the construction of a pseudorandom generator from a one-way permutation. The security reduction shows how to use any adversary A that is able to distinguish the output of the generator from a truly random string with probability $1/2 + \epsilon$ in order to construct a machine A' that inverts a one-way function with probability polynomial in ϵ. As in the case of simulations, the overhead of the reduction is measured as the ratio between the *worst-case* running time of A and the expected time of A'. We wish to argue that also in this case, the relevant measure should compare the expected running-time of A and A'. Indeed,

> *Many attacks on cryptographic primitives are probabilistic and their success probability is usually related to their* expected running-time *(and not their worst-case running-time).*

At first sight the difference between expected and worst-case running-time might not seem like a serious concern for the case of reductions, as any machine A with some expected running-time can be turned into a machine $\tilde{A}$ with a comparable running-time and comparable success probability, by simply truncating the execution of A after an appropriate number of steps and relying on the Markov inequality.[7] Note, however, that such a "truncation" significantly worsens the quality of the reduction (and in particular its concrete security) – in fact, a very "efficient" reduction might become completely impractical when considering the expected running-time of an adversary!

For concreteness, assume we have an *optimal* (in terms of concrete security) reduction from a pseudorandom generator G to some specific one-way function f, i.e., any adversary A with running-time t that breaks the pseudorandom generator with probability ϵ (i.e., it distinguishes the output of the generator G from a random string with probability $1/2+\epsilon$ can be turned into a machine A' with

[6] Our results are actually a bit stronger; see Appendix 4.3 for more details.

[7] The reason that such a truncation cannot be performed in the case of a zero-knowledge simulation is that a truncated simulator no longer is guaranteed to output a view that is indistinguishable from the real view of the verifier.

running-time t that inverts f with probability ϵ. Consider an adversary A with *expected* running-time t, that breaks the pseudorandom generator with probability ϵ. By the Markov inequality we get that the machine $\tilde{A}$, obtained by truncating the execution of A after $2t/\epsilon$ steps breaks the pseudorandom generator with probability $\epsilon/2$. By relying on our magical optimal reduction, $\tilde{A}$ can be used to invert f with probability $\epsilon/2$ in time $2t/\epsilon$. In essence, the truncation has turned an optimal reduction into one with quadratic overhead!

Our results show that for "natural" reductions, truncation is not needed. In particular, this establishes that all "natural" reductions with "good" concrete security, still have "good" concrete security when considering the expected running-time of an attacker. In other words, our results establish that:

> *Security proofs using "natural" reductions, provide the (roughly) the "same" concrete security with respect to adversaries with expected running-time t, as those with worst-case running-time t.*

In particular, when considering the concrete security of a construction, it is sufficient to prove security (using a "natural" reduction) with respect to adversaries with some specific *worst-case* running time t — security with respect to adversaries with expected running-time t then directly follows by our results.

1.3 Preliminaries and Notation

Our probabilistic notation follows [20]; see appendix A.1 for more details. By algorithm, we mean a Turing machine. For simplicity, we only consider algorithms with finite (i.e., bounded) running-time. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, the notation "$M_r(x)$" denotes the output of the M on input x when receiving r as random tape. We let the notation "$M_\bullet(x)$" denote the probability distribution over the outputs of M on input x where each bit of the random tape r is selected at random and independently, and then outputting $M_r(x)$ (note that this is a well-defined probability distribution since we only consider algorithms with finite running-time). Our protocol notation is described in Appendix A.2. We emphasize that in our notion of a *view* of an interaction, we only consider the part of the input and random tapes *actually* read by the parties.

When measuring the running-time of algorithms, for simplicity (and in accordance with the literate, see e.g., [12]), we assume that an algorithm M, given the code of a second algorithm A and an input x, can emulate the computation of A on input x with no (or linear) overhead.

Other preliminaries, such as the definition of indistinguishability of ensembles, interactive proofs, zero-knowledge proofs, knowledge tightness and precise $\mathcal{ZK}$ can be found in Appendix B.

1.4 Overview

Sections 2, 3 contain our results on precise encryption and proofs of knowledge; constructions of precise proofs of knowledge can be found in Appendix C. In section 4.2 we provide definitions of higher expected and higher-moment precision, and also provide upper-bounds for expected precision and higher-moment precision. In Appendix 5 we provide our results on secure computation.

2 Precise Encryption

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

Definition 1 (Encryption Scheme) *A triple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme, if* Gen *and* Enc *are p.p.t. algorithms and* Dec *is a deterministic polynomial-time algorithm,*

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

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

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

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

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

Definition 2 (Precise Encryption) *We say that the encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly/statistically/computationally secure with precision* p *under* ATK*, if, for every PPT ITM* A*, there exists a probabilistic algorithm* S *such that for every non-uniform PPT* M*, the following holds.*

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

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

where S_r *denotes the machine* S *with random tape fixed to* r*.*

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

Our main result is stated below.

Theorem 1 *If π is an CPA-secure public-key encryption scheme that on input a message of length n outputs a* pseudo-random *ciphertext of length $l(n)$, and furthermore the length $l(n)$ can be computed in time $O(n)$, then π is (computationally) secure with precision $p(n,t) = O(t)$ under CPA.*

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

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

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

Using the same proof we also get:

Theorem 2 *The one-time pad is perfectly secure encryption scheme with precision $p(n,t) = O(t)$ under CPA.*

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

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

3 Precise Proofs of Knowledge

[26] provided a technical definition of proof of knowledge aimed towards facilitating constructions of precise $\mathcal{ZK}$ proof. In essence, they 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. This notion is called *precisely emulatable proof of knowledge* in [29], and is similar in spirit to the notion of *witness extended emulation* [24].

We here provide a semantical notion of a proof of knowledge aimed a capturing the intuition that whenever a prover is able to convince the verifier, the prover must "know" a witness to the

statement; "know" is here capture by requiring that a witness can be computed in time closely related to the actual running-time of the prover in the convincing view.

In other words, we would like to have a notion of a proof of knowledge which requires that a prover can only convince the verifier whenever it "essentially" has a witness on its worktape at the end of the interaction. 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 3 (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*

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

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 3 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.

Constructions of precise proofs of knowledge We note that the precisely emulatable proofs of knowledge protocols of [26] are also precise proofs of knowledge; very similar proof techniques can be used to show this. For completeness, we provide a self-contained (and slightly simpler) proof in Section C.

4 Expected and Higher-moment Precision

4.1 Definitions

We put forward the notion of expected precision for zero-knowledge proofs.

Definition 4 (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 $x \in L, y \in R_L(x), z \in \{0,1\}^*$,*

$$\mathbf{Exp}[\mathsf{steps}_{S_\bullet(x,z)}] \leq p(|x|, T_{x,y,z})$$

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

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).[8] We obtain the notion of i'th moment precise $\mathcal{ZK}$ by exchanging condition 2 in Definition 4 for the following condition:

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

$$\mathbf{Exp}[(\mathsf{steps}_{S_\bullet(x,z)})^i] \leq p(|x|, T^{(i)}_{x,y,z})$$

where $T^{(i)}_{x,y,z}$ 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.

[8] 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 Upper Bounds on Expected Precision

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 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 [22] 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 5 (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.*

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

where $\texttt{query}_{S_{r'}^{V'_r(x,z)}(x)}(j)$ 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 $view$.

Remark: Our definition of a "natural" simulator is similar to the definition of a *poly*-strong black-box simulator of Katz and Lindell [22] (which is a special-case of their notion of an α-strong simulator). Indeed, both definitions capture the intuitive requirement that the simulator should feed its oracle messages that are indistinguishable from real prover messages. We mention, however, that our definition is somewhat different and significantly simpler than that of [22] (this is of course due to the different nature of the applications of it).

Theorem 3 *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 $E^{real}_{V'}$ 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 $E^{sim}_{V'}$. To simplify notation, we let

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

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

$$E^{sim}_{V'} = \frac{1}{2^R} \sum_{r \in \{0,1\}^R} \sum_{j}^{\infty} \Big[\mathsf{steps}_{V'}(q_r(j))\Big] = \sum_{j}^{\infty} \mathbf{Exp}\Big[\mathsf{steps}_{V'}(q_\bullet(j))\Big] \tag{1}$$

Expanding the above expression, we get

$$E^{sim}_{V'} = \sum_{j}^{\infty} \sum_{i}^{k(n)} \Pr\Big[|q_\bullet(j))| = i\Big] \mathbf{Exp}\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v)\Big]$$

We bound the second factor

$$\begin{aligned}
&\mathbf{Exp}\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v)\Big] = \\
&\sum_{t}^{\infty} \Pr\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v) \geq t\Big] = \\
&\sum_{t}^{\mathsf{time}_{V'}(|x|)} \Pr\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v) \geq t\Big]
\end{aligned}$$

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

$$\begin{aligned}
&\Big|\Pr\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v) \geq t\Big] - \\
&\Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \;:\; \mathsf{steps}_{V'}(v^i) \geq t\Big]\Big| \leq \mu(|x|)
\end{aligned}$$

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

$$\begin{aligned}
&\sum_{t}^{\mathsf{time}_{V'}(|x|)} \Pr\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v) \geq t\Big] - \\
&\sum_{t}^{\mathsf{time}_{V'}(|x|)} \Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \;:\; \mathsf{steps}_{V'}(v^i) \geq t\Big] \leq \\
&\sum_{t}^{\mathsf{time}_{V'}(|x|)} \mu(|x|) = \mu'(|x|)
\end{aligned}$$

In other words,

$$\begin{aligned}\mathbf{Exp}\Big[v \leftarrow q_\bullet(j) \;\Big|\; |v| = i \;:\; \mathsf{steps}_{V'}(v)\Big] - \\ \mathbf{Exp}\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \;:\; \mathsf{steps}_{V'}(v^i)\Big] \leq \\ \mu'(|x|)\end{aligned}$$

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

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

Plugging this in to equation 1 we get

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

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,

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

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 [21], we note that the theorem 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 [15, 11, 14, 4] are natural, we directly get:

Theorem 4 *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 5 *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 $p(n,t)$.*

4.3 Upper Bounds on Higher-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 $\texttt{round}_n(\cdot)$ such that on input a statement $x \in \{0,1\}^n$, the j'th query of the simulator is always a partial transcript containing $i = \texttt{round}_n(j)$ 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 6 (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 $\texttt{round}_n : N \to [k(n)]$, 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.*

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

where $\texttt{query}_{S_{r'}^{V'_r(x,z)}(x)}(j)$ 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 $view$.

Theorem 6 *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 Theorem 3, to simplify notation, we let

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

By definition,

$$M_S^{(m)} = \mathbf{Exp}\Big[(\mathsf{steps}_{S_\bullet^{V'_r(x,z)}})^m\Big] \leq \mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \Big(p_1(|x|) + \sum_j^{p_2(|x|)} \mathsf{steps}_{V'}(v_j)\Big)^m\Big]$$

(We emphasize that in the above expression the queries $v_1, v_2, ...$ 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[9] and by the monotonicity of expectations, we get

$$M_S^{(m)} \leq \mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \Big(p_2(|x|)+1\Big)^{m-1}\Big(p_1(|x|)^m + \sum_j^{p_2(|x|)} \mathsf{steps}_{V'}(v_j)^m\Big)\Big] =$$

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

By linearity of expectation, we get

$$M_S^{(m)} \leq \Big(p_2(|x|)+1\Big)^{m-1}\Big(p_1(|x|)^m + \sum_j^{p_2(|x|)} \mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m\Big]\Big)$$

We next show, in the same way as in the proof of Theorem 3 that there exists a negligible function $\mu(n)$ such that

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

As in Theorem 3,

$$\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m\Big] =$$

$$\sum_t^\infty \Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m \geq t\Big] =$$

$$\sum_t^{\mathsf{time}_{V'}(|x|)^m} \Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m \geq t\Big]$$

[9]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 $(\sum_i |x_i|)^m \leq (\sum_i |x_i|^m)(\sum_i 1)^{m-1}$.

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

$$\begin{aligned}
&\Pr\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m \geq t\Big] \leq \\
\Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] : \mathsf{steps}_{V'}(v^{\mathtt{round}_{|x|}(j)})^m \geq t\Big] + \mu'(|x|) &\leq \\
&\Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \ : \ \mathsf{steps}_{V'}(v)^m \geq t\Big] + \mu'(|x|)
\end{aligned}$$

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

$$\begin{aligned}
&\mathbf{Exp}\Big[v_j \leftarrow q_\bullet(j) \ : \ \mathsf{steps}_{V'}(v_j)^m\Big] \leq \\
\mu(|x|) + \sum_{t}^{\mathsf{time}_{V'}(|x|)^m} \Pr\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] : \mathsf{steps}_{V'}(v)^m \geq t\Big] &= \\
\mu(|x|) + \mathbf{Exp}\Big[v \leftarrow \text{VIEW}_2[P_\bullet(x,y) \leftrightarrow V'_r(x,z)] \ : \ \mathsf{steps}_{V'}(v)^m\Big] &\leq \\
M^{(m)}_{V'} + \mu(|x|)
\end{aligned}$$

Plugging this into equation 2 and continuing, we get

$$\begin{aligned}
M^{(m)}_S \leq \Big(p_2(|x|)+1\Big)^{m-1} \cdot \Big(p_1(|x|)^m + \sum_{j}^{p_2(|x|)} (M^{(m)}_{V'} + \mu(|x|)\Big) \leq \\
\Big(p_2(|x|)+1\Big)^{m} M^{(m)}_{V'} + \Big(p_2(|x|)+1\Big)^{m} p_1(|x|)^m + \mu''(|x|)
\end{aligned}$$

for some negligible function μ''. ■

By observing that the strict polynomial-time simulators of the protocols of [15, 6, 4] are natural, we directly get:

Theorem 7 *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 8 *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 Theorem 6 can be useful in the construction of such protocols, as Barak and Lindell [1] show that only languages in $\mathcal{BPP}$ have constant-round $\mathcal{ZK}$ arguments with *strict* polynomial-time black-box simulators. (Note, however, that Theorem 6 still can be useful also when considering non-interactive or constant-round primitives: many traditional *reductions* have strict polynomial running-time; consequently, Theorem 6 can be used to deduce an expected precision for such reductions.)

Second-moment precision of constant-round zero-knowledge protocols. We mention that the simulators for the classical constant-round protocols of Feige and Shamir [11] and Goldreich and Kahan [14] 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.

4.3.1 Precise $\mathcal{ZK}$ v.s. Higher-Moment Precise $\mathcal{ZK}$

We end this section by noting that precise $\mathcal{ZK}$ protocols are also higher-moment precise. We start by showing that the notion of precise $\mathcal{ZK}$ guarantees that the actual running-time distribution of the verifier is respected by the simulator.

Precise $\mathcal{ZK}$ preserves running-time distribution. 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, $p : N \times N \rightarrow N$ 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 $t \in N$:*

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

where

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

and

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

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

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

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

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

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

By using exactly the same proof we also get:

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

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

where

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

and

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

Precise $\mathcal{ZK}$ preserves higher-order moments. We show that for all protocols that are $\mathcal{ZK}$ with precision $p(n,t) = p'(n)t$, it holds that for all PPT 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 μ 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 $M^{(i)}_{V'}(x,z)$ 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 $M^{(i)}_S(x,z)$ denote the i'th moment of the running-time distribution of $S_\bullet(x,z)$. We start by noting that

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

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

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

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

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

$$M^{(i)}(x,z) \leq \sum_{t}^{\mathsf{time}_{V'}(|x|)^i} \Big(\Pr\Big[\frac{\mathsf{steps}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\Big] + \mu(|x|)\Big) =$$

$$\mu(|x|) \cdot \mathsf{time}_{V'}(|x|)^i + \sum_{t}^{\mathsf{time}_{V'}(|x|)^i} \Big(\Pr\Big[\frac{\mathsf{steps}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\Big]\Big)$$

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

$$M^{(i)}(x,z) \leq \mu'(|x|) + \sum_{t}^{\mathsf{time}_{V'}(|x|)^i} \Pr\Big[\frac{\mathsf{steps}_{S_\bullet(x,z)}}{p'(|x|)} \geq t^{1/i}\Big]$$

$$= \mu'(|x|) + \sum_{t}^{\mathsf{time}_{V'}(|x|)^i} \Pr\Big[\frac{(\mathsf{steps}_{S_\bullet(x,z)})^i}{p'^i(|x|)} \geq t\Big] =$$

$$\mu'(|x|) + \frac{1}{p'^i(|x|)} \sum_{t}^{\mathsf{time}_{V'}(|x|)^i} \Pr\Big[S_\bullet(x,z)^i \geq t\Big] =$$

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

■

5 Precise Secure Computation

We provide a precise analogue of the notion of secure computation [15]. We consider a malicious *static* computationally bounded (i.e., probabilistic polynomial-time) adversary that is allowed to corrupt a $t(m)$ of the m parties—that is, before the beginning of the interaction the adversary corrupts a subset of the players that may deviate arbitrarily from the protocol. The parties are assumed to be connected through authenticated point-to-point channels. For simplicity we here assume that the channels are synchronous.

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

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

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

Each party P_i receives it input x_i from the input vector $\bar{x} = (x_1, .., x_m)$. Each honest party P_i sends x_i to the trusted party. Corrupted parties P_i can instead send any arbitrary value $x'_i \in \{0,1\}^{|x_i|}$ to the trusted party. When the trusted third party has received messages x'_i from all parties (both honest and corrupted) it sets $\bar{x}' = (x'_1, .., x'_m)$, computes $y_i = f(\bar{x}')$ and sends y_i to P_i for every $i \in I$. When the adversary sends the message (send-output, i) to the trusted party, the trusted party delivers y_i to P_i. Finally, each honest party P_i outputs the value y_i received by the trusted party. The corrupted parties, and also S, may output any arbitrary value.

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

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

Definition 7 (Precise Secure Computation) *Let f and Π be as above, and $p : N \times N \times N \to N$ a monotonically increasing function. Protocol Π is said to t-*securely compute *f with precision p and* perfect/statsitical/computational security *if for every probabilistic polynomial-time real-model adversary A, there exists an probabilistic ideal-model adversary S, such that for every $m \in N$, $\bar{x} = (x_1, ..., x_m) \in (\{0,1\}^*)^m$ and every $I \subset [m]$, with $|I| \leq t$, the following conditions hold:*

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

$$\Pr\left[view \leftarrow \mathsf{view}_{f,I,S}(n, \bar{x}, z) : \mathsf{steps}_S(view) \leq p(|x|, \mathsf{steps}_A(S(view))\right].$$

Theorem 9 *Assume the existence of enhanced trapdoor permutations. Then, for any m-party functionality f there exists a protocol Π that $(m-1)$-securely computes f with precision $p(n) = O(n)$ and computational security.*

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

1. We first show how to construct a precise secure computation protocol when having access to an *idealized* $\mathcal{ZK}$ proof of membership functionality [25, 28]. We here rely on the protocols of [15, 7, 1, 28] and on padding. This is possible since given access to a idealized $\mathcal{ZK}$ functionality, we can construct a straight-line simulatable secure computation protocol (see [28]); this means that the overhead $o(t, n)$ of the simulator is some fixed polynomial $p(n)$ only in the security parameter n, independent of the running-time of the verifier t (plus the linear time needed to simulate the adversary). Now, if we pad all protocol messages with $1^{p(n)}$, we can make sure that the adversary spends at least $p(n)$ steps before it will see any "real" protocol messages. Thus, we get linear precision.

2. In the next step we simply implement the idealized $\mathcal{ZK}$ proof of membership functionality with a precise $\mathcal{ZK}$ protocol.

■

By appropriately modifying the protocols of [2] and [30] (using padding), we instead get

Theorem 10 *For any m-party functionality f there exists a protocol Π that $(m/2-1)$-securely computes f with precision $p(n) = O(n)$ and statistical security (assuming a broadcast channel). For any m-party functionality f there exists a protocol Π that $(m/3-1)$-securely computes f with precision $p(n) = O(n)$ and perfect security.*

These protocol additionally satisfies a stronger definition, where *fairness* is guaranteed; we omit the definition and refer the reader to [13].

6 Acknowledgments

We wish to thank Sasha Devore for helpful discussions which lead to the study of higher-moment precision, and Vinod Vaikuntanathan and abhi shelat for their generous help.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

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

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

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

[22] 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.

[23] L. Levin. Average-Case Complete Problems. SIAM Journal of Computing, 1986, Vol. 15, pages 285-286.

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

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

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

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

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

[29] R. Pass. A Precise Computational Approach to Knowledge. Ph.D Thesis, MIT, June 2006.

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

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

A Preliminaries

A.1 General Notation

We employ the following general notation.

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

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

Probabilistic notation. We employ the following probabilistic notation from [20]. We focus on probability distributions $X : S \to R^+$ over finite sets S.

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

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

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

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

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

Algorithms. We employ the following notation for algorithms.

Deterministic algorithms. By an algorithm we mean a Turing machine. We only consider *finite* algorithms, i.e., machines that have some fixed upper-bound on their running-time (and thus always halt). If M is a deterministic algorithm, we denote by $\mathsf{steps}_{M(x)}$ the number of computational steps taken by M on input x. We say that an algorithm M has time-complexity $\mathsf{time}_M(n) = t(n)$, if $\forall x \in \{0,1\}^*\ \mathsf{steps}_{M(x)} \leq t(|x|)$. (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. For simplicity (and in accordance with the literate), 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 at no cost.

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

A.2 Protocol Notation

We assume familiarity with the basic notions of an *Interactive Turing Machine* [19] (ITM for brevity) and a *protocol*. Briefly, an ITM is a Turing Machine with a read-only *input* tape, a read-only *auxiliary input* tape, a read-only *random* tape, a read/write *work-tape*, a read-only communication tape (for receiving messages) a write-only communication tape (for sending messages) and finally an *output* tape. The content of the input (respectively auxiliary input) tape of an ITM A is called *the input* (respectively *auxiliary input*) *of* A and the content of the output tape of A, upon halting, is called *the output of* A.

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

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 $M_A = \{m^1_A, m^2_A, ...\}$, $M_B = \{m^1_B, m^2_B, ...\}$ 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^i_A being the i'th message *read* by A and in m^i_B being the i'th message *read* by B, and r_1, r_2 are the parts of the random tapes consumed by A and B respectively. We also denote such an execution by $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$.

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

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

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

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

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

Time Complexity of ITMs. We say that an ITM A has time-complexity $\mathsf{time}_A(n) = t(n)$, 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 upper-bound on the running time of A *independently* of the content of the messages it receives. In other words, the time complexity of A is the *worst-case* running time of A in *any* interaction.

B Basic Notions

B.1 Indistinguishability

The following definition of (computational) indistinguishability originates in the seminal paper of Goldwasser and Micali [17].

Definition 8 (Indistinguishability) *Let X and Y be countable sets. Two ensembles $\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be* computationally indistinguishable over X*, if for every probabilistic "distinguishing" algorithm D whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $x \in X, y \in Y$:*

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

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

B.2 Interactive Proofs and Arguments

We state the standard definitions of interactive proofs (introduced by Goldwasser, Micali and Rackoff [19]) and arguments (introduced by Brassard, Chaum and Crepeau [6]).

Definition 9 (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*

$$\Pr\left[\text{OUT}_2[P_\bullet(x,y) \leftrightarrow V_\bullet(x)] = 1\right] = 1$$

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

$$\Pr\left[\text{OUT}_2[P_\bullet(x,y) \leftrightarrow V_\bullet(x)] = 1\right] \leq \nu(|x|)$$

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.*

B.3 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 [18, 19]. We present their definition below.[10]

Definition 10 ($\mathcal{ZK}$) *Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P, V) an interactive proof (argument) system for L. We say that (P, V) is* perfect/statistical/computational $\mathcal{ZK}$*, if for every probabilistic polynomial-time interactive machine V' there exists a probabilistic algorithm S whose expected running-time is polynomial in the length of its first input, such that the following ensembles are identical/statistically close/computationally indistinguishable over L.*

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

[10] The definition we present here appears in the journal version [19] or [18]. It differs from the original definition of [18] 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 [19] mention that this refinement was independently suggested by the Oren [27], Tompa and Woll [31] and the authors.

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 [**?**]) 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 [12])

Definition 11 (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.*

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

Knowledge Tightness. Goldreich, Micali and Wigderson [15], and more recently [12] 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 [12].[11] More precisely,

Definition 12 *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 10) such that for all sufficiently long $x \in L$ and every $z \in \{0,1\}^*$ we have*

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

where $\mathsf{time}_{V'(x,z)}$ 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.

[11] To be precise, the authors of [15] 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 [15] 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 [12], 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.

B.4 Precise Zero-Knowledge

We recall the definition of precise $\mathcal{ZK}$ [26, 29] (originally called *local* $\mathcal{ZK}$ in [26]).

Definition 13 (Perfect Precise $\mathcal{ZK}$ [26]) *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 $r \in \{0,1\}^*$, $\mathsf{steps}_{S_r(x,z)} \leq p(|x|, \mathsf{steps}_{V'}(S_r(x,z)))$.*

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.

C Constructions of Precise Proofs of Knowledge

We start by recalling the notion of *special-sound* proofs [8]. (we mention that e.g., the protocol of Blum is known to be special-sound.) Intuitively, a three-round public-coin interactive proof is said to be special-sound, if a valid witness to the statement x can be readily computed from any two accepting proof-transcripts of x which have the same first message but different second messages. More generally, a k-round public-coin interactive proof is said to be special-sound if the $k-1$'st round is a verifier-round i (i.e., a round where the verifier is supposed to send a message) and a valid witness to the statement x can be readily computed from any two accepting proof-transcripts of x which have the same first $k-2$ messages but different $k-1$'st message.

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

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

Definition 14 (Special-sound Proof) *Let (P, V) be a k-round public-coin interactive proof for the language $L \in \mathcal{NP}$ with witness relation R_L. We say that the protocol (P, V) is* special sound *with respect to R_L, if the $k-1$'st-round of (P, V) is a verifier-round and there exits a polynomial-time extractor machine X, such that for all $x \in L$ and all consistent transcripts T_1, T_2 it holds that if $\text{ACCEPT}_V(x, T_1) = 1$, $\text{ACCEPT}_V(x, T_1) = 1$ then $X(T_1, T_2, x) \in R_L(x)$.*

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

We will require the use of special-sound proofs for which extraction can be performed "very" efficiently. We say that (P, V) is special-sound with linear extraction, if the predicate ACCEPT_V can be computed in linear time (in its inputs length) and the extractor X in definition 14 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 [8, **?**]. We here show that appropriate sequential repetition of a special-sound proof results in an precise proof of knowledge.

C.1 Linear precision using $\omega(\log n)$ rounds

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

Lemma 1 (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.[12]

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

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

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

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

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

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

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

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

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

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

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

[12] Note that alhough our non-black-box usage is quite "mild" in the sense that we only count the number of computational steps of adversary, we actually use code of the adversary to "learn" something "useful" about it. Previous non-black box reductions (c.f. [**?**]) only use the code of the adversary in order to "diagonalize" against it.

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

Using a similar proof, we also get:

Lemma 2 (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.*

See [29] for a complete proof.

D CCA-secure Precise Encryption: A "feasibility" result

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

Theorem 11 *Assume the existence of CCA2-secure encryption schemes. Then there exist a CCA2-secure encryption scheme with precision $p(n) = O(n)$.*

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

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

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

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

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

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