APPENDIX C

Corrections and Additions to Volume 1

There is no 100% guarantee in the world;
whoever wants 100% guarantee should not build anything.
Eng. Isidor Goldreich (1906–1995)

In this appendix, we list a few corrections and additions to the previous chapters of this work (which appeared in [108]).

C.1. Enhanced Trapdoor Permutations

Recall that a collection of trapdoor permutations, as defined in Definition 2.4.5, is a collection of permutations, $\{f_\alpha\}_\alpha$, accompanied by four probabilistic polynomial-time algorithms, denoted I, S, F, and B (for *index*, *sample*, *forward*, and *backward*), such that the following (syntactic) conditions hold:

1. On input 1^n, algorithm I selects a random n-bit long index α of a permutation f_α, along with a corresponding trapdoor τ;
2. On input α, algorithm S *samples* the domain of f_α, returning an almost uniformly distributed element in it;
3. For x in the domain of f_α, given α and x, algorithm F returns $f_\alpha(x)$ (i.e., $F(\alpha, x) = f_\alpha(x)$);
4. For y in the range of f_α if (α, τ) is a possible output of $I(1^n)$, then, given τ and y, algorithm B returns $f_\alpha^{-1}(y)$ (i.e., $B(\tau, y) = f_\alpha^{-1}(y)$).

The hardness condition in Definition 2.4.5 refers to the difficulty of inverting f_α on a uniformly distributed element of its range, when given only the range element and α. That is, let $I_1(1^n)$ denote the first element in the output of $I(1^n)$ (i.e., the index); then for every probabilistic polynomial-time algorithm A (resp., every non-uniform family of

polynomial-size circuit $A = \{A_n\}_n$), every positive polynomial p, and all sufficiently large n's,

$$\mathsf{Pr}[A(I_1(1^n), f_{I_1(1^n)}(S(I_1(1^n))) = S(I_1(1^n))] < \frac{1}{p(n)} \tag{C.1}$$

Namely, A (resp., A_n) fails to invert f_α on $f_\alpha(x)$, where α and x are selected by I and S as here. An equivalent way of writing Eq. (C.1) is

$$\mathsf{Pr}[A(I_1(1^n), S'(I_1(1^n), R_n)) = f^{-1}_{I_1(1^n)}(S'(I_1(1^n), R_n))] < \frac{1}{p(n)} \tag{C.2}$$

where S' is the residual two-input (deterministic) algorithm obtained from S when treating the coins of the latter as an auxiliary input, and R_n denotes the distribution of the coins of S on n-bit long inputs. That is, A fails to invert f_α on x, where α and x are selected as earlier.

Although this definition suffices for many applications, in some cases we will need an enhanced hardness condition. Specifically, we will require it to be hard to invert f_α on a random input x (in the domain of f_α), *even when given the coins used by S in the generation of x*. (Note that given these coins (and the index α), the resulting domain element x is easily determined.)

Definition C.1.1 (enhanced trapdoor permutations): *Let $\{f_\alpha : D_\alpha \to D_\alpha\}$ be a collection of trapdoor permutations as in Definition 2.4.5. We say that this collection is* enhanced (and call it an enhanced collection of trapdoor permutations) *if for every probabilistic polynomial-time algorithm A, every positive polynomial p, and all sufficiently large n's,*

$$\mathsf{Pr}[A(I_1(1^n), R_n) = f^{-1}_{I_1(1^n)}(S'(I_1(1^n), R_n))] < \frac{1}{p(n)} \tag{C.3}$$

where S' is as in the foregoing discussion. The non-uniform version is defined analogously.

We comment that the RSA collection (presented in Section 2.4.3.1 and further discussed in Section 2.4.4.2) is, in fact, an *enhanced* collection of trapdoor permutations,[1] provided that RSA is hard to invert in the same sense as assumed in Section 2.4.3.1. In contrast, the Rabin Collection (as defined in Section 2.4.3) does not satisfy Definition C.1.1 (because the coins of the sampling algorithm give away a modular square root of the domain element). Still, the Rabin Collection can be easily modify to yield an *enhanced* collection of trapdoor permutations, provided that factoring is hard (in the same sense as assumed in Section 2.4.3). Actually, we present

[1] Here and in the following, we assume that sampling Z_N^*, for a composite N, is trivial. However, sampling Z_N^* (or even Z_N) by using a sequence of unbiased coins is not that trivial. The straightforward sampler may take $\ell \stackrel{\text{def}}{=} 2\lfloor \log_2 N \rfloor$ random bits, view them as an integer in $i \in \{0, 1, ..., 2^\ell - 1\}$, and output $i \bmod N$. This yields an almost uniform sample in Z_N. Also note that given an element $e \in Z_N$, one can uniformly sample an $i \in \{0, 1, ..., 2^\ell - 1\}$ such that $i \equiv e \pmod N$. Thus, the actual sampler does not cause trouble with respect to the enhanced hardness requirement.

two such possible modifications:

1. *Modifying the functions*. Rather than squaring modulo the composite N, we consider the function of raising to the power of 4 modulo N. It can be shown that the resulting permutations over the quadratic residues modulo N satisfy Definition C.1.1, provided that factoring is hard. Specifically, given N and a random $r \in Z_N$, the ability to extract the 4th root of r^2 mod N (modulo N) yields the ability to factor N, where the algorithm is similar to the one used in order to establish the intractability of extracting square roots.
2. *Changing the domains*. Rather than considering the permutation induced (by the modular squaring function) on the set Q_N of the quadratic residues modulo N, we consider the permutations induced on the set M_N, where M_N contains all integers in $\{1, ..., N/2\}$ that have Jacobi symbol modulo N that equals 1. Note that as in the case of Q_N, each quadratic residue has a unique square root in M_N (because exactly two square roots have a Jacobi symbol that equals 1 and their sum equals N).[2] However, unlike Q_N, membership in M_N can be determined in polynomial-time (when given N without its factorization). Thus, sampling M_N can be done in a straightforward way, which satisfies Definition C.1.1.

   Actually, squaring modulo N is a 1-1 mapping of M_N to Q_N. In order to obtain a permutation over M_N, we modify the function a little, such that if the result of modular squaring is bigger than $N/2$, then we use its additive inverse (i.e., rather than outputting $y > N/2$, we output $N - y$).

We comment that the special case of Definition 2.4.5 in which the domain of f_α equals $\{0, 1\}^{|\alpha|}$ is a special case of Definition C.1.1 (because, without loss of generality, the sampling algorithm may satisfy $S'(\alpha, r) = r$). Clearly, the RSA and the Rabin collections can be slightly modified to fit the former special case.

Correction to Volume 1. Theorems 4.10.10, 4.10.14, and 4.10.16 (which in turn are based on Remark 4.10.6) refer to the existence of certain non-interactive zero-knowledge proofs. The claimed non-interactive zero-knowledge proof systems can be constructed by assuming the existence of an *enhanced* collection of trapdoor permutations. However, in contrast to the original text, it is not known how to derive these proof systems based on the existence of a (regular) collection of trapdoor permutations. See further discussion in Section C.4.1.

Open Problem. *Is it possible to convert any collection of trapdoor permutations into an enhanced one?* An affirmative answer will resolve open problems stated in Sections 7.7.6 and C.4.1, which refer to the assumptions required for General Secure Multi-Party Computation and various types of Non-Interactive Zero-Knowledge proofs, respectively.

[2] As in the case of Q_N, we use the fact that -1 has Jacobi symbol 1.

C.2. On Variants of Pseudorandom Functions

The focus of Section 3.6 was on a special case of pseudorandom functions, hereafter referred to as the fixed-length variant. For some function $\ell : \mathbb{N} \to \mathbb{N}$ (e.g., $\ell(n) = n$), these functions map $\ell(n)$-bit long strings to $\ell(n)$-bit long strings, where n denotes the lengths of the function's seed. More general definitions were presented in Section 3.6.4. In particular, functions mapping strings of *arbitrary length* to $\ell(n)$-bit long strings were considered. Here, we refer to the latter as the variable-length variant.

A natural question regarding these variants is how to *directly* (or efficiently) transform functions of the fixed-length variant into functions of the variable-length variant.[3] Exercises 30 and 31 in Chapter 3 *implicitly suggest such a transformation,* and so does Proposition 6.3.7. Because of the interest in this natural question, we next state the actual result explicitly.

Proposition C.2.1: *Let* $\{f_s : \{0,1\}^{\ell(|s|)} \to \{0,1\}^{\ell(|s|)}\}_s$ *be a* (fixed-length) *pseudorandom function ensemble, and* $\{h_r : \{0,1\}^* \to \{0,1\}^{\ell(|r|)}\}_r$ *be a generalized hashing ensemble with a* $(t, 1/t)$*-collision property,*[4] *for some super-polynomial function* $t : \mathbb{N} \to \mathbb{N}$. *Then* $\{g_{s,r} = f_s \circ h_r\}_{s,r:|s|=|r|}$ *is a* (variable-length) *pseudorandom function ensemble.*

Proof Idea: The proofs of Propositions 6.3.6 and 6.3.7 actually establish Proposition C.2.1. ■

Comment. Alternative constructions of variable-length pseudorandom functions based on fixed-length pseudorandom functions are presented in [25, 22, 13]. In these works, the fixed-length pseudorandom functions are applied to each block of the input, and so the number of applications is linearly related to the input length (rather than being a single one). On the other hand, these works do not use variable-length hashing. Indeed, these works presuppose that a fixed-length pseudorandom function (rather than a variable-length one) is non-expensive (and, in practice, is available as an off-the-shelf product).

C.3. On Strong Witness Indistinguishability

Unfortunately, we have to withdraw two claims regarding *strong* witness indistinguishable proofs as defined in Definition 4.6.2.[5] Specifically, in general, *strong* witness

[3] An indirect construction may use the fixed-length variant in order to obtain a one-way function, and then construct the variable-length variant using this one-way function. Needless to say, this indirect construction is very wasteful.

[4] Recall that the $(t, 1/t)$-collision property means that for every $n \in \mathbb{N}$ and every $x \neq y$ such that $|x|, |y| \leq t(n)$, the probability that $h_r(x) = h_r(y)$ is at most $1/t(n)$, where the probability is taken over all possible choices of $r \in \{0,1\}^n$ with uniform probability distribution.

[5] We comment that the notion of *strong* witness indistinguishability was introduced by the author at a late stage of writing [108].

indistinguishability is not closed under parallel composition (and so Lemma 4.6.7 is wrong). Consequently, contrary to what is stated in Theorem 4.6.8, we do not know whether there exist constant-round public-coin proofs with negligible error that are *strong* witness indistinguishable for languages out of $\mathcal{BPP}$.[6] Before discussing the reasons for withdrawing these claims and the consequences of doing so, we stress that the flaws pointed out here only refer to *strong* witness indistinguishability and not to (regular) witness indistinguishability. That is, as stated in Lemma 4.6.6, (regular) witness indistinguishability is closed under parallel composition, and thus the part of Theorem 4.6.8 that refers to regular witness indistinguishability is valid (i.e., providing constant-round public-coin proofs with negligible error that are witness indistinguishable for $\mathcal{NP}$).

Notation. To facilitate the rest of the discussion, we let WI stand for "(regular) witness indistinguishability" and strong-WI stand for "strong witness indistinguishability."

C.3.1. On Parallel Composition

A counter-example to Lemma 4.6.7 can be derived by using the protocol presented at the end of Section 4.5.4.1 (and assuming the existence of one-way functions); that is, this protocol is (zero-knowledge and hence) strong-WI, but executing it twice in parallel (on the same common input) is not strong-WI. Tracing the error in the reasoning outlined in Section 4.6.2, we stress a fundamental difference between WI and strong-WI. Under the former (i.e., under the definition of WI), the indistinguishability of executions, in which the prover uses one out of two possible NP-witnesses (for the same common input), holds even when the (adversary) verifier is given these two NP-witnesses. The analogous claim does not necessarily hold for strong-WI, because these two NP-witnesses (even presented in random order) may allow for distinguishing one possible common input from the other (provided that these two possibilities are *not* identical, unlike in the case of WI). Now, observe that the single-session adversary constructed in the proof of Lemma 4.6.6 needs to get the NP-witnesses that correspond to the other sessions in order to emulate these sessions. However, these other NP-witnesses may determine the two possible NP-witnesses for the current session, and so the indistinguishability of the executions of the current session is no longer guaranteed. Furthermore, the other NP-witnesses may even uniquely determine the NP-witness (or the input triple) used in the current session. Indeed, the source of trouble is in the possible dependence among the NP-witnesses used in the various sessions. Consequently, we can resurrect parallel compositions (of strong-WI) for the special case in which the NP-witnesses used in the various sessions are independently distributed. Actually, we need statistical independence among the (entire) input triples used in the various sessions.

Lemma C.3.1 (Parallel Composition for Strong Witness Indistinguishability, Revisited): *Let $L \in \mathcal{NP}$, R_L, (P, V), Q, R_L^Q, and P_Q be as in Lemma 4.6.6, and suppose*

[6] Theorem 4.6.8 does not mention the public-coin condition, but the construction that is supposed to support it is of the public-coin type. Note that constant-round zero-knowledge protocols are presented in Section 4.9, but these are in relaxed models and are not of the public-coin type.

that (P, V) is strong witness indistinguishable. Then for every two probability ensembles $\{(\overline{X}^1_n, \overline{Y}^1_n, \overline{Z}^1_n)\}_{n\in\mathbb{N}}$ and $\{(\overline{X}^2_n, \overline{Y}^2_n, \overline{Z}^2_n)\}_{n\in\mathbb{N}}$ such that $\overline{X}^j_n = (X^j_{n,1}, ..., X^j_{n,Q(n)})$, $\overline{Y}^j_n = (Y^j_{n,1}, ..., Y^j_{n,Q(n)})$, and $\overline{Z}^j_n = (Z^j_{n,1}, ..., Z^j_{n,Q(n)})$, where $(X^j_{n,i}, Y^j_{n,i}, Z^j_{n,i})$ is independent of $(X^\ell_{n,k}, Y^\ell_{n,k}, Z^\ell_{n,k})_{k\neq i,\ell\in\{1,2\}}$, the following holds:

> *If $\{(\overline{X}^1_n, \overline{Z}^1_n)\}_{n\in\mathbb{N}}$ and $\{(\overline{X}^2_n, \overline{Z}^2_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable, then so are $\{\langle P_Q(\overline{Y}^1_n), V^*_Q(\overline{Z}^1_n)\rangle(\overline{X}^1_n)\}_{n\in\mathbb{N}}$ and $\{\langle P_Q(\overline{Y}^2_n), V^*_Q(\overline{Z}^2_n)\rangle(\overline{X}^2_n)\}_{n\in\mathbb{N}}$, for every probabilistic polynomial-time machine V^*_Q.*

We stress that the components of $\overline{Y}^j_n$ (resp., $\overline{Z}^j_n$) may depend on the corresponding components of $\overline{X}^j_n$, but they are independent of the other components of $\overline{Y}^j_n$ (resp., $\overline{Z}^j_n$), as well as of the other components of $\overline{X}^j_n$. Note that statistical independence of this form holds vacuously in Lemma 4.6.6, which refers to fixed sequences of strings. Lemma C.3.1 is proved by extending the proof of Lemma 4.6.6. Specifically, we consider hybrids as in the original proof, and construct a verifier V^* that interacts with P on the i-th session (or copy), while emulating all the other sessions (resp., copies). Toward this emulation, we provide V^* with the corresponding $Q(n) - 1$ components of both $\overline{Y}^j_n$'s (as well as of both $\overline{X}^j_n$'s and $\overline{Z}^j_n$'s). Fixing the best possible choice for these $Q(n) - 1$ components, we derive a verifier that interacts with P and contradicts the hypothesis that (P, V) is strong witness indistinguishable. The key point is that revealing (or fixing) the other $Q(n) - 1$ components of both $\overline{Y}^j_n$'s does not allow for distinguishing the i-th component of $\overline{X}^1_n$ and $\overline{Z}^1_n$ from the i-th component of $\overline{X}^2_n$ and $\overline{Z}^2_n$.

C.3.2. On Theorem 4.6.8 and an Afterthought

Unfortunately, Theorem 4.6.8 is proved by a parallel composition that refers to the same common input (and the same NP-witness). Thus, Lemma C.3.1 is not applicable, and consequently we do not know whether the part of Theorem 4.6.8 that refers to *strong* witness indistinguishable proofs is valid (when referring to public-coin proofs). This is indeed an interesting open problem.

We comment that one can reduce the construction of constant-round (public-coin) *strong* witness indistinguishable proofs with negligible error for $\mathcal{NP}$ to the construction of such proofs for the special case in which the two X^j_n's (and Y^j_n's) are identically distributed (and the Z^j_n's are only computationally indistinguishable). Consider, for example, the following protocol:

1. The prover sends a commitment to the value 0.

2. Using a (regular) witness indistinguishable proof (as provided by Theorem 4.6.8), the prover proves that either the common input is in the language or the string sent at Step 1 is a commitment to 1.
Let us denote by T^j_n the transcript of the execution of this step, when the common input is X^j_n (and the parties use auxiliary inputs Y^j_n and Z^j_n, respectively). It can

be proven that the T_n^j's are computationally indistinguishable (by considering what happens if at Step 1 the prover sends a commitment to 1).

3. Using a *strong* witness indistinguishable proof (which is indeed the missing component or the sub-protocol to which the current protocol is reduced), the prover proves that the string sent at Step 1 is a commitment to 0.
   Note that it suffices to show that the verifier cannot distinguish the two possible transcript distributions of the current step, where both possible distributions refer to executions with the same common input (i.e., the commitment) and the same prover's auxiliary input (i.e., the decommitment information). In contrast, these two distributions (of executions) refer to two different distributions of the verifier's auxiliary input (i.e., either T_n^1 or T_n^2), which are indistinguishable.

The foregoing reduction demonstrates that the notion of strong witness indistinguishability actually refers to issues that are fundamentally different from witness indistinguishability. Specifically, the issue is whether or not the interaction with the prover helps to distinguish between two possible distributions of some auxiliary information (which are indistinguishable without such an interaction). Furthermore, this issue arises also in case the prover's auxiliary inputs (i.e., the "witnesses") are identically distributed.

C.3.3. Consequences

In view of the fact that we do not have constant-round public-coin strong witness indistinguishable proofs with negligible error for $\mathcal{NP}$, we suggest replacing the use of such proofs with some cumbersome patches. A typical example is the construction of non-oblivious commitment schemes (i.e., Theorem 4.9.4).

Non-Oblivious Commitment Schemes. We begin the discussion by noting that the specific formulation appearing in Definition 4.9.3 is wrong. One should partition the commit phase into two sub-phases, such that the second sub-phase is a proof-of-knowledge of the input and coins used by the sender at the first sub-phase, which in turn should constitute (by itself) a commitment scheme. That is, the view in the relation displayed in Definition 4.9.3 should be the view of the first sub-phase (rather than the view of the entire commit phase). In fact, for the *current implementation,* we need a relaxed definition in which one only proves knowledge of the input (but not of the coins) used by the sender at the first sub-phase. We stress that the input value proved to be known must be such that it is impossible for the sender to later decommit to a different value. Indeed, in the relaxed form, we do not require a later decommitment to be at all possible; we only require that if decommitment takes place, then the outcome should match the said input value. Note that this relaxed form suffices for the proof presented in Section 4.9.2.2.

Next, we modify the construction used in the proof of Theorem 4.9.4 as follows. First, rather than sending one ordinary commitment to the input, we send many such (independent) commitments. Secondly, rather than using a (constant-round) proof-of-knowledge with negligible error, we use one that has constant error. The point is that

such a (constant-round) proof-of-knowledge that is zero-knowledge (and, hence, strong witness indistinguishable) is known. We invoke this proof system many times, in parallel, where each invocation is applied to a different commitment. Thus, we can apply Lemma C.3.1 and conclude that these executions are strong witness indistinguishable (where the witnesses are the coins used in the ordinary commitments), and therefore, the entire protocol constitutes a (complicated) commitment scheme. Finally, one can establish the non-oblivious property by using the knowledge-extractor associated with the proof system. Note that we can only extract the committed input and part of the coins used at the first stage (i.e., the coins used in some of the ordinary commitments but not necessarily the coins used in all of them). Furthermore, it may be that we also accept in case the sequence of strings sent at the first stage does not correspond to any legitimate sequence (i.e., of commitments to the same value). However, if we extract one value, then it is impossible for the sender to later decommit to a different value, because the extracted value always fits at least one of the individual commitments.

Other Applications. Fortunately, Theorem 4.9.4 is the only place where *strong* witness indistinguishable proofs are used in this work. We believe that in many other applications of *strong* witness indistinguishable proofs, an analogous modification can be carried out (in order to salvage the application). A typical example appears in [7]. Indeed, the current situation is very unfortunate, and we hope that it will be redeemed in the future. Specifically, we propose the following open problem:

Open Problem. *Construct constant-round* public-coin *strong witness indistinguishable proofs* (and proofs-of-knowledge) *with negligible error for* $\mathcal{NP}$, or prove that this cannot be done. Recall that zero-knowledge *arguments* of this nature are known [5]. The challenge is in providing such *proofs*.

C.4. On Non-Interactive Zero-Knowledge

In retrospect, it appears that Section 4.10 is too laconic. As is usually the case, laconic style gives rise to inaccuracies and gaps, which we wish to address here. (See also Section C.6.)

C.4.1. On NIZKs with Efficient Prover Strategies

In continuation of Remark 4.10.6 and following [32], we briefly discuss the issues that arise when we wish to implement Construction 4.10.4 by an efficient prover. Recall that Remark 4.10.6 outlines such an implementation, while using a family of trapdoor permutations of the form $\{f_\alpha : \{0,1\}^{|\alpha|} \to \{0,1\}^{|\alpha|}\}_{\alpha \in \overline{I}}$, where the index-set $\overline{I}$ is efficiently recognizable. Unfortunately, no family of trapdoor permutations of this particular form (and, in particular, with an efficiently recognizable $\overline{I}$) is known. Thus, we first extend the treatment to the case in which $\overline{I}$ is not necessarily efficiently recognizable. The problem we encounter is that the prover may select (and send) a function that is not in

the family (i.e., an α not in $\overline{I}$). In such a case, the function is not necessarily 1-1, and, consequently, the soundness property may be violated. This concern can be addressed by using a (simple) non-interactive (zero-knowledge) proof for establishing that the function is "typically 1-1" (or, equivalently, is "almost onto the designated range"). The proof proceeds by presenting pre-images (under the function) of random elements specified in the reference string. Note that for any fixed polynomial p, we can only prove that the function is 1-1 on at least a $1 - (1/p(n))$ fraction of the designated range (i.e., $\{0,1\}^n$), yet this suffices for moderate soundness of the entire proof system (which in turn can be amplified by repetitions). For further details, consult [32].

Although the known candidate trapdoor permutations can be modified to fit this form, we wish to further generalize the result such that any *enhanced* trapdoor permutation (as in Definition C.1.1) can be used. This can be done by letting the reference string consist of the coin sequences used by the domain-sampling algorithm (rather than of elements of the function's domain). By virtue of the enhanced hardness condition (i.e., Eq. (C.3)), the security of the hard-core is preserved, and so is the zero-knowledge property.

As stated at the end of Section C.1, in contrast to what was claimed in Remark 4.10.6, we do not known how to extend the construction to arbitrary (rather than enhanced) trapdoor permutations. This leads to the following open problem.

Open Problem. *Under what intractability assumptions is it possible to construct non-interactive zero-knowledge proofs with efficient prover strategies for any set in $\mathcal{NP}$? In particular, does the existence of arbitrary collections of trapdoor permutations suffice?* We comment that the assumption used here affects the assumption used in (general) constructions of public-key encryption schemes that are secure under chosen ciphertext attacks (see, e.g., Theorem 5.4.31).

C.4.2. On Unbounded NIZKs

The preliminary discussion is Section 4.10.3.1 reduces the general treatment to a treatment of assertions of a priori bounded length, but the former is not defined formally. To close this gap, we note that a definition that covers assertions of a priori unbounded length can be derived from Definition 4.10.11 by considering inputs in $\cup_{i=1}^{\mathrm{poly}(n)} L_i$, rather than in L_{n^ε}. In view of the key role of efficient provers in this setting, it is also adequate to present a definition that covers this aspect. This can be done analogously to the formulations used in the following Proposition C.4.1.

The proof of Proposition 4.10.13 relies on the fact that witness indistinguishability of non-interactive protocols is preserved under parallel composition *even if the same reference string is used in all copies*. That is, we claim and use the following result (where R is typically an NP-relation):

Proposition C.4.1: *Let P be a probabilistic polynomial-time algorithm such that for every infinite sequence of triples of the form $\bar{t} \stackrel{\mathrm{def}}{=} (x,u,v)$, where $(x,u),(x,v) \in R$, it holds that $\{(U_{\mathrm{poly}(|x|)}, P(x,u,U_{\mathrm{poly}(|x|)}))\}_{\bar{t}}$ and $\{(U_{\mathrm{poly}(|x|)}, P(x,v,U_{\mathrm{poly}(|x|)}))\}_{\bar{t}}$*

are computationally indistinguishable.[7] *Then for every polynomial p and every infinite sequence of sequences of the form $\bar{s} \stackrel{\text{def}}{=} (x_1, ..., x_t, u_1, ..., u_t, v_1, ..., v_t)$, where $n \stackrel{\text{def}}{=} |x_1| = \cdots = |x_t|$, $t \stackrel{\text{def}}{=} p(n)$ and $(x_j, u_j), (x_j, v_j) \in R$ for $j = 1, .., t$, it holds that the ensembles $\{(U_{\text{poly}(n)}, P(x_1, u_1, U_{\text{poly}(n)}), ..., P(x_t, u_t, U_{\text{poly}(n)}))\}_{\bar{s}}$ and $\{(U_{\text{poly}(n)}, P(x_1, v_1, U_{\text{poly}(n)}), ..., P(x_t, v_t, U_{\text{poly}(n)}))\}_{\bar{s}}$ are computationally indistinguishable.*

We stress that the same reference string (i.e., $U_{\text{poly}(n)}$) is used in all invocations of the prover P. Thus, Proposition C.4.1 does not refer to multiple samples of computationally indistinguishable ensembles (nor even to independent samples from a sequence of computationally indistinguishable pairs of ensembles, as would have been the case if the various invocations were to use independently distributed reference strings). Still, Proposition C.4.1 can be established by using the hybrid technique. The key observation is that, given a single proof with respect to some reference string along with the reference string (as well as the relevant sequence $\bar{s}$), one can efficiently generate all the other proofs (with respect to the same reference string). Indeed, the *internal coins* used by P in each of these proofs are independent.

C.4.3. On Adaptive NIZKs

In Definition 4.10.15, the *adaptive zero-knowledge* condition should be quantified only over efficiently computable input-selection strategies. Furthermore, it seems that also the witness-selection strategies should be restricted to ones implemented by polynomial-size circuits. The revised form is presented in Definition 5.4.22.

A few words regarding the proof of Theorem 4.10.16 seem appropriate. The (two-stage) simulation procedure itself is sketched in footnote 29 (of Chapter 4). Recall that at the first stage, we generate matrices at random, and replace the useful matrices with all-zero matrices (i.e., matrices of f-images that have pre-images with hard-core value equal to zero). In the second stage, when given an adaptively chosen graph, we reveal all elements of all non-useful matrices and the required elements of the useful matrices (i.e., the non-edges), where revealing an element means revealing the corresponding f-pre-image. In establishing the quality of this simulation procedure, we rely on the hypothesis that the input graph, as well as a Hamiltonian cycle in it, are determined by a polynomial-size circuit.[8] Loosely speaking, assuming toward the contradiction that the simulation can be distinguished from the real proof, we construct a circuit that distinguishes a sequence of random $f(x)$'s with $b(x) = 0$ from a sequence of random $f(x)$'s with $b(x) = 1$. This "b-value distinguisher" places the tested f-images in the suitable entries (i.e., those corresponding to the predetermined Hamiltonian cycles) of useful matrices, fills up the rest of the entries of the useful matrices with elements it generates in $\{f(x) : b(x) = 0\}$, and fills the entries of non-useful matrices with random f-images that it generates (conditioned on their yielding non-useful matrices). We stress

[7] Recall that the distinguisher is also given the index of the distribution, which in this case is the triple $\bar{t}$.

[8] Indeed, here is where we use the fact that the corrected definition (see Definition 5.4.22) refers only to input-selection and witness-selection strategies that can be implemented by polynomial-size circuits.

that the simulator generates f-images by selecting random pre-images and applying f to each of them, and so it knows the pre-images and can reveal them later. Next, the simulator determines the input graph and the corresponding Hamiltonian cycle (by using the abovementioned polynomial-size circuit) and acts as the real prover. Finally, it feeds the original distinguisher with the corresponding output. Observe that in case the given sequence of $f(x)$'s satisfies $b(x) = 0$ (resp., $b(x) = 1$) for each $f(x)$, the "b-value distinguisher" produces outputs distributed exactly as in the simulation (resp., the real proof).

C.5. Some Developments Regarding Zero-Knowledge

A recent result by Barak [5] calls for reevaluation of the significance of all negative results regarding black-box zero-knowledge[9] (as defined in Definition 4.5.10). In particular, relying on standard intractability assumptions, Barak presents round-efficient public-coin zero-knowledge arguments for $\mathcal{NP}$ (using non-black-box simulators), whereas only $\mathcal{BPP}$ can have such *black-box* zero-knowledge arguments (see comment following Theorem 4.5.11). It is interesting to note that Barak's simulator works in strict (rather than expected) probabilistic polynomial-time, addressing an open problem mentioned in Section 4.12.3. Barak's result is further described in Section C.5.2

In Section C.5.1, we review some recent progress in the study of the preservation of zero-knowledge under concurrent composition. We seize the opportunity to provide a wider perspective on the question of the preservation of zero-knowledge under various forms of protocol composition operations.

We mention that the two problems discussed in this section (i.e., the "preservation of security under various forms of protocol composition" and the "use of the adversary's program within the proof of security") arise also with respect to the security of other cryptographic primitives. Thus, the study of zero-knowledge protocols serves as a good benchmark for the study of various problems regarding cryptographic protocols.

C.5.1. Composing Zero-Knowledge Protocols

A natural question regarding zero-knowledge proofs (and arguments) is whether or not the zero-knowledge condition is preserved under a variety of composition operations. Three types of composition operation were considered in the literature: *sequential composition*, *parallel composition,* and *concurrent composition*. We note that the preservation of zero-knowledge under these forms of composition not only is interesting for its own sake but also sheds light on the preservation of the security of general protocols under these forms of composition.

We stress that when we talk of the composition of protocols (or proof systems), we mean that the honest users are supposed to follow the prescribed program (specified in the protocol description) that refers to a single execution. That is, the actions of

[9] Specifically, one should reject the interpretation, offered in Section 4.5 (see Sections 4.5.0, 4.5.4.0, and 4.5.4.2), by which negative results regarding black-box zero-knowledge indicate the inherent limitations of zero-knowledge.

honest parties in each execution are independent of the messages they received in other executions. The adversary, however, may coordinate the actions it takes in the various executions, and in particular, its actions in one execution may also depend on messages it received in other executions.

Let us motivate the asymmetry between the postulate that honest parties act independently in different executions and the absence of such an assumption with respect to the adversary's actions. Typically, coordinating actions in different executions is difficult but not impossible. Thus, it is desirable to use stand-alone protocols that preserve security under "composition" (as defined earlier), rather than to use protocols that include inter-execution coordination actions. Note that at the very least, inter-execution coordination requires users to keep track of all executions that they perform. Actually, trying to coordinate honest executions is even more problematic than it seems, because one may need to coordinate executions of *different* honest parties (e.g., all employees of a big corporation or an agency under attack), which in many cases is highly unrealistic. On the other hand, the adversary attacking the system may be willing to go to the extra trouble of coordinating its attack in the various executions of the protocol.

For $T \in \{\texttt{sequential}, \texttt{parallel}, \texttt{concurrent}\}$, we say that a protocol is T-zero-knowledge if it is zero-knowledge under a composition of type T. The definitions of T-zero-knowledge are derived from the standard definition by considering appropriate adversaries (i.e., adversarial verifiers), that is, adversaries that can initiate a polynomial number of interactions with the prover, where these interactions are scheduled according to the type T.[10] The corresponding simulator (which, as usual, interacts with nobody) is required to produce an output that is computationally indistinguishable from the output of such a type T adversary.

C.5.1.1. Sequential Composition

Sequential composition refers to a situation in which the protocol is invoked (polynomially) many times, where each invocation follows the termination of the previous one. At the very least, security (e.g., zero-knowledge) should be preserved under sequential composition, or else the applicability of the protocol is highly limited (because one cannot safely use it more than once).

We mention that whereas the "simplified" version of zero-knowledge (i.e., without auxiliary inputs, as in Definition 4.3.2) is not closed under sequential composition (see [113]), the actual version (i.e., with auxiliary inputs, as in Definition 4.3.10) is closed under sequential composition (see Section 4.3.4). We comment that the same phenomenon arises when trying to use a zero-knowledge proof as a sub-protocol inside larger protocols. Indeed, it is for these reasons that the augmentation of the "basic" definition by auxiliary inputs was adopted in all subsequent works.[11]

[10] Without loss of generality, we may assume that the adversary never violates the scheduling condition; it may instead send an illegal message at the latest possible adequate time. Furthermore, without loss of generality, we may assume that all the adversary's messages are delivered at the latest possible adequate time.

[11] The preliminary version of Goldwasser, Micali, and Rackoff's work [124] uses the "basic" definition (i.e., Definition 4.3.2), whereas the final version of that work as well as most subsequent works use the augmented

C.5.1.2. Parallel Composition

Parallel composition refers to a situation in which (polynomially) many instances of the protocol are invoked at the same time and proceed at the same pace. That is, we assume a synchronous model of communication, and consider (polynomially) many executions that are totally synchronized, such that the i-th message in all instances is sent exactly (or approximately) at the same time. (Natural extensions of this model are discussed here as well as at the end of Section C.5.1.3.)

It turns out that, in general, zero-knowledge is not closed under parallel composition. A simple counter-example (to the "parallel composition conjecture") is outlined in Section 4.5.4.1 (following [113]). This counter-example consists of a simple protocol that is zero-knowledge (in a strong sense) but is not closed under parallel composition (not even in a very weak sense).[12]

We comment that in the 1980s, parallel composition was studied mainly in the context of *round-efficient error reduction* (cf. [91, 113]); that is, the aim was to construct full-fledged zero-knowledge proofs (with negligible soundness error) by composing (in parallel) a basic zero-knowledge protocol of high (but bounded away from 1) soundness error. Since alternative ways of constructing constant-round zero-knowledge proofs (and arguments) were found (cf. [112, 90, 47]), interest in parallel composition (of zero-knowledge protocols) has died. In retrospect, this was a conceptual mistake, because parallel composition (and mild extensions of this notion) capture the preservation of security in a fully synchronous (or almost fully synchronous) communication network. We note that the almost fully synchronous communication model is quite realistic in many settings, although it is certainly preferable not to assume even weak synchronism.

Although, in general, zero-knowledge is not closed under parallel composition, under standard intractability assumptions (e.g., the intractability of factoring), there exist zero-knowledge protocols for $\mathcal{NP}$ that are closed under parallel composition. Furthermore, these protocols have a constant number of rounds (cf. [109] for proofs and [82] for arguments).[13] Both results also extend to concurrent composition in a synchronous communication model, where the extension is in allowing protocol invocations to start at different times (and, in particular, executions may overlap but not run simultaneously).

We comment that parallel composition is also problematic in the context of reducing the soundness error of arguments (cf. [24]), but our focus here is on the zero-knowledge aspect of protocols, regardless of whether they are proofs, arguments, or neither.

C.5.1.3. Concurrent Composition (with and without Timing)

Concurrent composition generalizes both sequential and parallel composition. Here (polynomially) many instances of the protocol are invoked at arbitrary times and proceed

definition (i.e., Definition 4.3.10). In some works, the "basic" definition is used for simplicity, but typically one actually needs and means the augmented definition.

[12] The presentation in Section 4.5.4.1 is in terms of two protocols, each being zero-knowledge, such that executing them in parallel is not zero-knowledge. These two protocols can be easily combined into one protocol (e.g., by letting the second party determine, in its first message, which of the two protocols to execute).

[13] In the case of parallel zero-knowledge *proofs*, there is no need to specify the soundness error because it can always be reduced via parallel composition. As mentioned later, this is not the case with respect to arguments.

at an arbitrary pace. That is, we assume an asynchronous (rather than synchronous) model of communication.

In the 1990s, when extensive two-party (and multi-party) computations became a reality (rather than a vision), it became clear that it is (at least) desirable that cryptographic protocols maintain their security under concurrent composition (cf. [77]). In the context of zero-knowledge, concurrent composition was first considered by Dwork, Naor, and Sahai [82]. Actually, two models of concurrent composition were considered in the literature, depending on the underlying model of communication (i.e., a *purely asynchronous model* and an *asynchronous model with timing*).

Concurrent Composition in the Pure Asynchronous Model. Here we refer to the standard model of asynchronous communication. In comparison to the timing model, the pure asynchronous model is a simpler model, and using it requires no assumptions about the underlying communication channels. However, it seems harder to construct concurrent zero-knowledge protocols for this model. In particular, for a while it was not known whether concurrent zero-knowledge proofs for $\mathcal{NP}$ exist at all (in this model). Under standard intractability assumptions (e.g., the intractability of factoring), this question was affirmatively resolved by Richardson and Kilian [175]. Following their work, research has focused on determining the round-complexity of concurrent zero-knowledge proofs for $\mathcal{NP}$. Currently, this question is still open, and the state of the art regarding it is as follows:

- Under standard intractability assumptions, every language in $\mathcal{NP}$ has a concurrent zero-knowledge proof with *almost logarithmically* many rounds (cf. [169], building upon [138], which in turn builds over [175]). Furthermore, the zero-knowledge property can be demonstrated by using a black-box simulator (see the definition in Section 4.5.4.2 and the discussion in Section C.5.2).
- Black-box simulators cannot demonstrate the concurrent zero-knowledge property of non-trivial proofs (or arguments) having significantly less than logarithmically many rounds (cf. Canetti et al. [58]).[14]
- Recently, Barak [5] demonstrated that the "black-box simulation barrier" can be bypassed. With respect to concurrent zero-knowledge, he obtained only the following partial result: Under standard intractability assumptions, every language in $\mathcal{NP}$ has a constant-round zero-knowledge argument (rather than proof) that maintains security as long as an a priori bounded (polynomial) number of executions take place concurrently. (The length of the messages in his protocol grows linearly with this a priori bound.)

Thus, it is currently unknown whether or not *constant-round* arguments for $\mathcal{NP}$ may be concurrent zero-knowledge (in the pure asynchronous model).

[14] By *non-trivial* proof systems we mean ones for languages outside $\mathcal{BPP}$, whereas by *significantly less than logarithmic* we mean any function $f:\mathbb{N}\to\mathbb{N}$ satisfying $f(n)=o(\log n/\log\log n)$. In contrast, by *almost logarithmic* we mean any function f satisfying $f(n)=\omega(\log n)$.

Concurrent Composition under the Timing Model. A model of naturally limited asynchronousness (which certainly covers the case of parallel composition) was introduced by Dwork, Naor, and Sahai [82]. Essentially, they assume that each party holds a local clock such that the relative clock rates are bounded by an a priori known constant, and they consider protocols that employ time-driven operations (i.e., `time-out` incoming messages and `delay` outgoing messages). The benefit of the timing model is that it seems easier to construct concurrent zero-knowledge protocols for it. Specifically, using standard intractability assumptions, *constant-round* arguments and proofs that are concurrent zero-knowledge under the timing model do exist (cf. [82] and [109], respectively). The disadvantages of the timing model are discussed next.

The timing model consists of the *assumption* that talking about the actual timing of events is meaningful (at least in a weak sense) and of the *introduction of time-driven operations*. The timing assumption amounts to postulating that each party holds a local clock and knows a global bound, denoted $\rho \geq 1$, on the relative rates of the local clocks.[15] Furthermore, it is postulated that the parties know a (pessimistic) bound, denoted Δ, on the message-delivery time (which also includes the local computation and handling times). In our opinion, these timing assumptions are most reasonable, and are unlikely to restrict the scope of applications for which concurrent zero-knowledge is relevant. We are more concerned about the effect of the time-driven operations introduced in the timing model. Recall that these operations are the `time-out` of incoming messages and the `delay` of outgoing messages. Furthermore, *typically* the delay period is at least as long as the time-out period, which in turn is at least Δ (i.e., the time-out period must be at least as long as the pessimistic bound on message-delivery time so as not to disrupt the proper operation of the protocol). This means that the use of these time-driven operations yields a slowing down of the execution of the protocol (i.e., running it at the rate of the pessimistic message-delivery time, rather than at the rate of the actual message-delivery time, which is typically much faster). Still, in the absence of more appealing alternatives (i.e., a constant-round concurrent zero-knowledge protocol for the pure asynchronous model), the use of the timing model may be considered reasonable. (We comment than other alternatives to the timing model include various set-up assumptions; cf. [55, 72].)

Back to Parallel Composition. Given our opinion about the timing model, it is not surprising that we consider the problem of parallel composition almost as important as the problem of concurrent composition in the timing model. Firstly, it is quite reasonable to assume that the parties' local clocks have approximately the same rate, and that drifting is corrected by occasional clock synchronization. Thus, it is reasonable to assume that the parties have an approximately good estimate of some global time. Furthermore, the global time may be partitioned into phases, each consisting of a constant number of rounds, so that each party wishing to execute the protocol just delays its invocation to the beginning of the next phase. Thus, concurrent execution

[15] The rate should be computed with respect to reasonable intervals of time; for example, for Δ as defined next, one may assume that a time period of Δ units is measured as Δ' units of time on the local clock, where $\Delta/\rho \leq \Delta' \leq \rho\Delta$.

of (constant-round) protocols in this setting amounts to a sequence of (time-disjoint) almost parallel executions of the protocol. Consequently, proving that the protocol is parallel zero-knowledge suffices for concurrent composition in this setting.

Relation to Resettable Zero-Knowledge. Going to the other extreme, we mention that there exists a natural model of zero-knowledge that is even stronger than concurrent zero-knowledge (even in the pure asynchronous model). Specifically, "resettable zero-knowledge" as defined in [55] implies concurrent zero-knowledge.

C.5.2. Using the Adversary's Program in the Proof of Security

Recall that the definition of zero-knowledge proofs states that whatever an efficient adversary can compute after interacting with the prover can be efficiently computed from scratch by a so-called *simulator* (which works without interacting with the prover). Although the simulator may depend arbitrarily on the adversary, the need to present a simulator for each feasible adversary seems to require the presentation of a universal simulator that is given the adversary's strategy (or program) as another auxiliary input. The question addressed in this section is how the universal simulator can use the adversary's program.

The adversary's program (or strategy) is actually a function that determines for each possible view of the adversary (i.e., its input, random choices, and the message it has received so far) which message will be sent next. Thus, we identify the adversary's program with this next-message function. As stated previously, until very recently, all universal simulators (constructed toward demonstrating zero-knowledge properties) have used the adversary's program (or rather its next-message function) as a black-box (i.e., the simulator invoked the next-message function on a sequence of arguments of its choice). Furthermore, in view of the presumed difficulty of "reverse-engineering" programs, it was commonly believed that nothing is lost by restricting attention to simulators, called black-box simulators, that only make black-box usage of the adversary's program. Consequently, Goldreich and Krawczyk conjectured that impossibility results regarding black-box simulation represent inherent limitations of zero-knowledge itself, and studied the limitations of the former [113].

> In particular, they showed that parallel composition of the protocol of Construction 4.4.7 (as well as of any constant-round public-coin protocol) *cannot be proven to be zero-knowledge using a black-box simulator,* unless the language (i.e., 3-Colorability) is in $\mathcal{BPP}$. In fact, their result refers to any constant-round public-coin protocol with negligible soundness error, regardless of how such a protocol is obtained. This result was taken as strong evidence toward the conjecture that a constant-round public-coin protocol with negligible soundness error *cannot be zero-knowledge* (unless the language is in $\mathcal{BPP}$).
>
> Similarly, as mentioned in Section C.5.1.3, it was shown that protocols of a sub-logarithmic number of rounds *cannot be proven to be concurrent zero-knowledge via a black-box simulator* [58]. Again, this was taken as evidence toward the conjecture that such protocols cannot be *concurrent zero-knowledge*.

In contrast to these conjectures (and to the reasoning underlying them), Barak showed how to construct non-black-box simulators and obtained several results that were known to be unachievable via black-box simulators [5]. In particular, under standard intractability assumptions (see also [7]), he presented constant-round public-coin zero-knowledge arguments with negligible soundness error for any language in $\mathcal{NP}$. (Moreover, the simulator runs in strict polynomial-time, which is impossible for black-box simulators of non-trivial constant-round protocols [9].) Furthermore, these protocols preserve zero-knowledge under a fixed[16] polynomial number of concurrent executions, in contrast to the result of [58] (regarding black-box simulators) that also holds in that restricted case. Thus, Barak's result calls for the reevaluation of many common beliefs. Most concretely, it says that results regarding black-box simulators do not reflect inherent limitations of zero-knowledge (but rather an inherent limitation of a natural way of demonstrating the zero-knowledge property). Most abstractly, it says that there are meaningful ways of using a program other than merely invoking it as a black-box.

Does this means that a method was found to "reverse-engineer" programs or to "understand" them? We believe that the answer is negative. Barak [5] is using the adversary's program in a significant way (i.e., more significant than just invoking it), without "understanding" it. *So, how does he use the program?*

The key idea underlying Barak's protocol [5] is to have the prover prove that either the original NP-assertion is valid or that he (i.e., the prover) "knows the verifier's residual strategy" (in the sense that it can predict the next verifier message). Indeed, in a real interaction (with the honest verifier), it is infeasible for the prover to predict the next verifier message, and so computational soundness of the protocol follows. However, a simulator that is given the code of the verifier's strategy (and not merely oracle access to that code) can produce a valid proof of the disjunction by properly executing the sub-protocol using its knowledge of an NP-witness for the second disjunctive. The simulation is computational indistinguishable from the real execution, provided that one cannot distinguish an execution of the sub-protocol in which one NP-witness (i.e., an NP-witness for the original assertion) is used from an execution in which the second NP-witness (i.e., an NP-witness for the auxiliary assertion) is used. That is, the sub-protocol should be a *witness indistinguishable* argument system (see Sections 4.6 and 4.8). We warn the reader that the actual implementation of this idea requires overcoming several technical difficulties (cf. [5, 7]).

Perspective. In retrospect, taking a wide perspective, it should not come as a surprise that the program's code yields extra power beyond black-box access to it. Feeding a program with its own code (or part of it) is the essence of the diagonalization technique, and this, too, is done without reverse engineering. Furthermore, various non-black-box techniques have appeared before in the cryptographic setting, but they were used in the more natural context of *devising an attack* on an (artificial) insecure scheme (e.g., toward

[16] The protocol depends on the polynomial that bounds the number of executions, and thus is not known to be concurrent zero-knowledge (because the latter requires fixing the protocol and then considering any polynomial number of concurrent executions).

proving the failure of the "Random Oracle Methodology" [54] and the impossibility of software obfuscation [8]). In contrast, in [5] (and [6]), the code of the adversary is being used within a sophisticated proof of security. What we wish to highlight here is that *non-black-box usage of programs is also relevant to proving* (rather than to disproving) *the security of systems*.

Digest: Witness Indistinguishability and the FLS-Technique

The foregoing description (of [5]), as well as several other sophisticated constructions of zero-knowledge protocols (e.g., [89, 175]), make crucial use of a technique introduced by Feige, Lapidot, and Shamir [89], which in turn is based on the notion of witness indistinguishability (introduced by Feige and Shamir [91]). This technique, hereafter referred to as the FLS-technique, was used in Construction 4.10.12, but we wish to further discuss it next.

Following is a sketchy description of a special case of the FLS-technique, whereas the abovementioned application uses a more general version (which refers to proofs-of-knowledge, as defined in Section 4.7).[17] In this special case, the technique consists of the following construction schema, which uses witness indistinguishable protocols for $\mathcal{NP}$ in order to obtain zero-knowledge protocols for $\mathcal{NP}$. On common input $x \in L$, where L is the NP-set defined by the witness relation R, the following two steps are performed:

1. The parties generate an instance x' for an auxiliary NP-set L', where L' is defined by a witness relation R'. The generation protocol in use must satisfy the following two conditions:
   - **(a)** If the verifier follows its prescribed strategy, then no matter which feasible strategy is used by the prover, with high probability, the protocol's outcome is a NO-instance of L'.
   - **(b)** There exists an efficient (non-interactive) procedure for producing a (random) transcript of the generation protocol *along with an NP-witness for the corresponding outcome* (which is a YES-instance of L'), such that the produced transcript is computationally indistinguishable from the transcript of a real execution of the protocol.
2. The parties execute a *witness indistinguishable* protocol for the set L'' defined by the witness relation $R'' = \{((u, u'), (v, v')) : (u, v) \in R \vee (u', v') \in R'\}$. The sub-protocol is such that the corresponding prover can be implemented in probabilistic polynomial-time, given an NP-witness for $(u, u') \in L''$. The sub-protocol is invoked on common input (x, x'), where x' is the outcome of Step 1, and the sub-prover

[17] In the general case, the generation protocol may generate an instance x' in L', but it is infeasible for the prover to obtain a corresponding witness (i.e., a w' such that $(x', w') \in R'$). In the second step, the sub-protocol in use ought to be a proof-of-knowledge, and computational soundness of the main protocol will follow (because otherwise, the prover, using a knowledge-extractor, can obtain a witness for $x' \in L'$).

is invoked with the corresponding NP-witness as auxiliary input (i.e., with (w, λ), where w is the NP-witness for x given to the main prover).

The computational soundness of this protocol follows by Property (a) of the generation protocol (i.e., with high probability $x' \notin L'$, and so $x \in L$ follows by the soundness of the protocol used in Step 2). To demonstrate the zero-knowledge property, we first generate a simulated transcript of Step 1 (with outcome $x' \in L'$), along with an adequate NP-witness (i.e., w' such that $(x', w') \in R'$), and then emulate Step 2 by feeding the sub-prover strategy with the NP-witness (λ, w'). Combining Property (b) of the generation protocol and the witness indistinguishability property of the protocol used in Step 2, the simulation is indistinguishable from the real execution.

C.6. Additional Corrections and Comments

Regarding Constriction 4.10.7 and the Proof of Proposition 4.10.9. The current description of the setting of the mapping of the input graph G to the Hamiltonian matrix H (via the two mappings π_1 and π_2) is confusing and even inaccurate. Instead, one may identify the rows (resp., columns) of H with $[n]$ and use a single permutation π over $[n]$ (which supposedly maps the vertices of G to those of H).[18] Alternatively, one may compose this permutation π with the two (1-1) mappings ϕ_i's (where $\phi_i : [n] \to [n^3]$ is as in the original text), and obtain related π_i's (i.e., $\pi_i(v) = \phi_i(\pi(v))$), which should be used as in the original text. We stress that the real prover determines π to be an isomorphism between the Hamiltonian cycle of G and the Hamiltonian cycle of H, whereas the simulator selects π at random.

Arguments-of-Knowledge. In continuation of Sections 4.7 and 4.9.2, we mention that the round-efficient argument system of [90] is actually an "argument-of-knowledge" (with negligible error). The interested reader is referred to [9] for further improvements regarding such proof systems. Essentially, using a relaxed (yet satisfactory) definition of an argument-of-knowledge, the latter work presents a constant-round zero-knowledge argument-of-knowledge with *strict* (rather than expected) probabilistic polynomial-time simulator and knowledge-extractor.

Some Missing Credits. The sequential composition lemma for zero-knowledge protocols (i.e., Lemma 4.3.11) is due to [119]. The notions of *strong* witness indistinguishability (Definition 4.6.2) and *strong* proofs-of-knowledge (Section 4.7.6), and the Hidden Bit Model (Section 4.10.2) have first appeared in early versions of this work.

[18] The identification is via the two mappings ϕ_1 and ϕ_2 mentioned in the original text. We stress that these mappings only depend on the matrix M that contains H.

C.7. Additional Mottoes

Motto for Section 3.2

Indistinguishable things are identical
(or should be considered as identical).

The Principle of Identity of Indiscernibles
G. W. Leibniz (1646–1714)

(Leibniz admits that counter-examples to this principle are conceivable but will not occur in real life because God is much too benevolent.)

Motto for Chapter 4

A: Please.
B: Please.
A: I insist.
B: So do I.
A: OK then, thank you.
B: You are most welcome.

A protocol for two Italians to pass through a door.
Source: Silvio Micali, 1985.

(The protocol is zero-knowledge because it can be simulated without knowing any of the secrets of these Italians; in fact, the execution is independent of their secrets as well as of anything else.)