J. Cryptology (1996) 9: 167–189

Journal of
CRYPTOLOGY

© 1996 International Association for
Cryptologic Research

How To Construct Constant-Round Zero-Knowledge Proof Systems for NP

Oded Goldreich
Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot, Israel

Ariel Kahan
BRM Technologies, P.O.Box 45065,
Jerusalem, Israel

Communicated by Gilles Brassard

Received 10 May 1993 and revised 8 March 1995

Abstract. Constant-round zero-knowledge proof systems for every language in $\mathcal{NP}$ are presented, assuming the existence of a collection of claw-free functions. In particular, it follows that such proof systems exist assuming the intractability of either the Discrete Logarithm Problem or the Factoring Problem for Blum integers.

Key words. Zero-knowledge proofs, NP, Claw-free functions, Commitment schemes, Coin flipping protocols, Parallel composition of proof systems.

1. Introduction

The applications of zero-knowledge proof systems to cryptography are too numerous and too well known to be listed here. We confine ourselves to pointing out two facts to which zero-knowledge proofs owe their wide applicability: firstly, the generality of the notion of zero-knowledge [11]; and, secondly, the ability to construct zero-knowledge proof systems for every NP statement (using a general intractability assumption) [10, Theorem 5]. However, to be of practical use, zero-knowledge proofs also have to be efficient.

A very important complexity measure for (cryptographic as well as arbitrary) protocols is their *round-complexity*. Namely, the number of message exchanges taking place in the course of the execution. The above quoted result of Goldreich *et al.* [10], by which the existence of one-way functions implies the existence of a zero-knowledge proof system for every language in $\mathcal{NP}$, is obtained using proof systems with very high round-complexity. Alternative constructions have lower, yet nonconstant, round-complexity. The purpose of this work is to present zero-knowledge proof systems, with *constant* round-complexity, for $\mathcal{NP}$.

1.1. *Clarifications*

A few clarifications are in order. First, we stress that by saying an *interactive proof system* we mean one with a **negligible error probability**. Sometimes, interactive proof systems are defined as having constant, say $1/3$, error probability. Such weak proof systems are of limited practical value on their own, and it is implicitly assumed that they are repeated sufficiently many times so that the error probability is reduced as desired. However, sequential repetitions of a protocol yield a corresponding increase in the round-complexity. In fact, in some sense, the problem addressed in this paper is how to reduce the error probability of weak interactive proofs, *without* increasing the round-complexity *and while preserving* their zero-knowledge property. Hence, for the sake of simplicity, we address the problem of constructing (constant-round) zero-knowledge proof systems with negligible error probability.[1]

We also stress that we consider interactive proof systems, as defined by Goldwasser *et al.* [11], rather than *computationally sound* proof systems (also known as *arguments*), as defined by Brassard *et al.* [4]. The difference between the two is sketched below. In (regular) interactive proof systems, the *soundness condition* requires that nobody, regardless of his/her computational abilities, can fool the verifier into accepting false statements (except with negligible probability). In *computationally sound* proof systems, the *soundness condition* refers only to computationally bounded cheating provers and, furthermore, it is typically proven to hold under some intractability assumption.

Finally, we stress that our approach depends, in an essential manner, on the standard definition of zero-knowledge which allows the simulator to run in **expected** polynomial time (see [11] and [10]). We do not know whether our results can be obtained under a more strict definition of zero-knowledge which only allows the simulator to run in (strict) polynomial time. We remark that many other popular results also depend on the same convention. For example, Graph Isomorphism (GI) is shown to have a perfect zero-knowledge proof using a simulator that runs for expected polynomial time [10, Theorem 2]. To the best of our knowledge, using a simulator that runs for strict polynomial time, it can only be shown that GI has an interactive proof which is almost-perfect (statistical) zero-knowledge. Even worse, Graph Non-Isomorphism is not known to have an almost-perfect zero-knowledge proof (under strict polynomial-time simulators), whereas it has a perfect zero-knowledge proof system [10, Theorem 3] (with respect to expected polynomial-time simulators).

1.2. *Our Main Result*

We show how to construct *constant-round* zero-knowledge interactive proof systems for any language in $\mathcal{NP}$. Our construction relies on the existence of collections of claw-free functions. Such functions exist if factoring Blum integers is hard (see [12]), or alternatively if the Discrete Logarithm Problem is intractable (see [3]).

[1] An alternative and more complex presentation is possible by considering the "knowledge tightness" of zero-knowledge proof systems with small (but nonnegligible) error probability. Loosely speaking, the knowledge tightness of a zero-knowledge protocol is an upper bound on the ratio between the running time of simulators for the protocol and the running time of the corresponding verifiers [10, Remark 18]. The aim is to construct constant-round proof systems with simultaneously small error probability and small knowledge tightness.

As usual in the area of zero-knowledge, the results are most simply stated using a nonuniform formalization. In this formalization the intractability assumptions are stated with respect to nonuniform families of polynomial-size circuits. A formalization in terms of uniform complexity is possible. See [7].

Remark. The work reported here has been cited in the literature already in 1988. However, no version of this work has ever appeared before.

1.3. *Related Work*

Constant-round zero-knowledge *computationally sound* proof (i.e., argument) systems for $\mathcal{NP}$ have been presented in [6] and [5]. As explained above, these protocols are weaker than ours in the sense that they do not constitute proof systems (with "unrestricted" soundness condition). However, these works also have advantages over ours. The advantage of the work of Feige and Shamir is that it uses a much weaker intractability assumption [6]; specifically, they only assume the existence of arbitrary one-way functions. The advantage of the work of Brassard *et al.* is that their protocol is *perfect* zero-knowledge [5], rather than just being computationally zero-knowledge.[2] (The intractability assumption in [5] is incomparable with ours, and seems stronger than the mere existence of one-way functions.) Hence, the three works (i.e., ours and those of [6] and [5]) are incomparable: each has some advantage over the other two.

Noninteractive zero-knowledge proof systems, as defined by Blum *et al.* [2], seem related to constant-round zero-knowledge proof systems. One has to be careful, though, and recall that in the setting of noninteractive proof systems both prover and verifier have access to a uniformly chosen string, called the *reference string*. We stress that the reference string is not selected by either parties, but is rather postulated to be uniformly chosen by some trusted third party. Clearly, combining a secure coin-flipping protocol (see [1]) with a noninteractive zero-knowledge proof system, a zero-knowledge proof system can be derived. Note, however, that the round-complexity of the resulting interactive proof system depends on the round-complexity of the coin-flipping protocol and on whether it can be securely performed *in parallel* many times. In fact, our work can be viewed as suggesting a coin-flipping protocol that remains secure even if executed in parallel polynomially many times.

Other efficiency measures related to zero-knowledge proofs and arguments have been investigated in many works; see, for example, [15] and [16].

1.4. *Organization*

We start with an overview of our approach and present an abstraction of a technical difficulty encountered and resolved. We then present the building blocks of our interactive proof system which are two "complementary" types of commitment schemes. A detailed description of our interactive proof system follows and we conclude by presenting a simulator which demonstrates that this interactive proof system is indeed zero-knowledge.

[2] The Feige–Shamir argument system, mentioned above, also has a perfect zero-knowledge version [6], but this version relies on seemingly stronger complexity theoretic assumptions than required for the computational zero-knowledge version.

2. Overview

We start by reviewing the standard zero-knowledge proof system for Graph 3-Colorability. This interactive proof system, presented by Goldreich *et al.* [10], proceeds by (a polynomial number of) *sequential* repetitions of the following **basic** protocol.

- *Common input*: A simple (3-colorable) graph $G = (V, E)$.
- *Prover's first step* (P1): Let ψ be a 3-coloring of G. The prover selects a random permutation, π, over $\{1, 2, 3\}$, and sets $\varphi(v) \stackrel{\text{def}}{=} \pi(\psi(v))$, for each $v \in V$. (Hence, the prover forms a random relabelling of the 3-coloring ψ.) The prover sends to the verifier a sequence of commitments so that the ith commitment is to the value $\varphi(i)$.
- *Verifier's first step* (V1): The verifier uniformly selects an edge $(i, j) \in E$, and sends it to the prover.
- *Motivating remark*: The verifier asks to inspect the colors of vertices i and j.
- *Prover's second step* (P2): The prover reveals the values corresponding to the ith and jth commitments.
- *Verifier's second step* (V2): The verifier accepts if and only if the revealed values are different elements of $\{1, 2, 3\}$ and if they indeed fit the corresponding commitments received in step P1.

It is shown in [10] that the basic protocol is zero-knowledge and that this property is preserved under *sequential* repetitions. Repetitions are required in order to reduce the error probability of the basic protocol, which might be as large as $1 - 1/|E|$, to a negligible function of $|G|$. However, sequential repetitions are out of the question if round-efficient protocols are sought. Hence, the key to round-efficient error reduction is parallel execution of the above basic protocol.[3] However, as demonstrated by Goldreich and Krawcyzk, the protocol which results from parallel execution of the basic protocol, sufficiently many times, cannot be proven zero-knowledge using a universal simulator which uses the verifier as a black-box [8]. We note that all known zero-knowledge protocols are proven to be zero-knowledge using such a universal simulator, and, furthermore, that it is hard to conceive an alternative way of proving that a protocol is zero-knowledge. Hence, slightly different approaches are required.

Two different approaches for resolving the above difficulties have been suggested in [10]. These two approaches share an underlying idea which is to let the verifier commit to its queries (i.e., a sequence of edges each corresponding to a different commitment to a coloring of the graph) *before* the prover commits to a sequence of colorings of the graph. The two approaches vary by the manner in which the verifier commits to its queries.

1. One possibility is to use an "ordinary" commitment scheme (like the one used by the prover).[4] This will enable a computationally unbounded prover to find out the queries before committing to the colorings, and thus cheat the verifier causing

[3] Namely, the prover independently generates many random relabelling of the coloring ψ and commits to each of them. The verifier then selects a query edge for each committed coloring, and checks the revealed colors supplied by the prover. If all fit the corresponding commitments and each pair of colors is different, then the verifier accepts.

[4] See Section 3 for a more formal discussion of various types of commitment schemes.

it to accept also graphs that are not 3-colorable. Yet, a computationally bounded cheating prover cannot break these commitments and hence the proposed protocol may be computationally sound.

2. The other possibility is to use a commitment scheme with perfect sęcrecy.[5] The disadvantage in this approach is that commitment schemes with perfect secrecy seem harder to construct that "regular" ones.

Implementing the two (above-mentioned) approaches turned out to be more difficult than anticipated. Nevertheless, the first approach has been implemented in [6] yielding zero-knowledge arguments for every language in $\mathcal{NP}$ provided that one-way functions exist. The current paper presents an implementation of the second approach.

The main difficulty in implementing the second approach is in the construction of the simulator demonstrating the zero-knowledge of the ("parallelized") interactive proof system sketched above. In the rest of this section we try to provide an abstract account of the difficulty and our approach to resolving it.

A Technical Problem Resolved

Preliminaries: We call a function $f\colon \mathsf{N} \mapsto \mathsf{R}$ *negligible* if, for every polynomial $P(\cdot)$ and all sufficiently large n's, $f(n) < 1/P(n)$. A function $f\colon \mathsf{N} \mapsto \mathsf{R}$ is called *nonnegligible* if a polynomial $P(\cdot)$ exists so that, for all sufficiently large n's, $f(n) > 1/P(n)$. Note that a function may be neither negligible nor nonnegligible. Both notions extend naturally to functions from strings to reals; for example, $F\colon \{0,1\}^* \mapsto \mathsf{R}$ is said to be negligible if $f(n) \stackrel{\text{def}}{=} \max_{x\in\{0,1\}^n} \{F(x)\}$ is negligible.

While constructing the zero-knowledge simulator for the "parallelized" interactive proof, a problem of the following nature arises. Suppose that we are given access to two probabilistic black-boxes denoted A and B. On input $x \in \{0,1\}^n$, the first black-box, A, outputs a "key" K with probability denoted $a(x)$ and halts without output otherwise (i.e., $a(x) \stackrel{\text{def}}{=} \text{Prob}(A(x) = 1)$). On input $x \in \{0,1\}^n$ and key K, the second black-box, B, produces an output (in $\{0,1\}^n$) with probability denoted $b(x)$ and otherwise halts with no output (or outputs the empty string λ). The absolute difference between $a(x)$ and $b(x)$ is negligible. We denote by $D(x)$ the output distribution $B(x, K)$ conditioned on $B(x, K) \neq \lambda$ (i.e., for every α, we have $\text{Prob}(D(x) = \alpha) = \text{Prob}(B(x, K) = \alpha | B(x, K) \neq \lambda)$). On input x, our goal is to output strings according to distribution $D(x)$ with probability at least $a(x)$ and otherwise indicate failure (say by outputting λ). Actually, we are allowed to output the strings according to $D(x)$ with probability which is at most negligibly smaller than $a(x)$. We are allowed to run in expected polynomial-time and invoke both black-boxes, where each invocation is charged at unit cost.

A natural attempt to solve the problem follows. On input x, we first invoke $A(x)$. If the output is not K, then we halt indicating failure, otherwise we repeatedly invoke $B(x, K)$ until a nonempty output is obtained. Clearly, the expected number of times that B is invoked is $a(x)/b(x)$. In case $a(x) \leq b(x)$ holds for all x's, this is OK. Another good

[5] Again, see Section 3.

case is when the function a: $\{0, 1\}^* \mapsto \mathsf{R}$ is nonnegligible (as in this case $a(x)/b(x)$ is very close to 1). We remark that in case the function a: $\{0, 1\}^* \mapsto \mathsf{R}$ is negligible we may always halt without output. The problem, however, is what to do in case the function a: $\{0, 1\}^* \mapsto \mathsf{R}$ is neither nonnegligible nor negligible and the ratio $a(x)/b(x)$ is not bounded by a polynomial (e.g., occasionally, $a(x) = 2^{-|x|}$ and $b(x) = 2^{-2|x|}$).

Our solution is slightly more complex. On input x, we first invoke $A(x)$ and *proceed only if* the output is K (otherwise we halt, indicating failure as before). Next, we approximate $a(x)$ by invoking $A(x)$ until we get output K for, say, $|x|^2$ times. This yields, with very high probability, an approximation of $a(x)$ up to a constant factor (i.e., the estimate is the ratio of $|x|^2$ over the number of invocations of $A(x)$). Denote this estimate by $\tilde{a}(x)$ and assume that $\tilde{a}(x) \geq 2^{-|x|}$ (otherwise set $\tilde{a}(x) = 2^{-|x|}$). We now invoke $B(x, K)$, for at most (say) $|x|^2/\tilde{a}(x)$ times, until a nonempty string is obtained. If such a string is obtained we output it, otherwise we halt with no output. Note that in case the first invocation of $A(x)$ outputs K we end up invoking the two black-boxes for $\mathrm{poly}(|x|)/a(x)$ times. Although $\mathrm{poly}(|x|)/a(x)$ may be more than a polynomial in $|x|$, its contribution to the *expected* running time is scaled down by a factor of $a(x)$, and so we obtain expected polynomial-time running time.

3. Commitment Schemes

Generally speaking commitment schemes are two-party protocols, partitioned into two phases, guaranteeing two conflicting requirements. The first phase, called *commit*, is supposed to commit the *sender* to a value without allowing the *receiver* to know which value this is. In the second phase, called *reveal*, the value determined by the first phase can be revealed. Hence, the conflicting requirements are *secrecy* of the value at the *commit phase* and *nonambiguity* of the value revealed later. These two conditions can be stated in information-theoretic or in computational terms. The information-theoretic formulation implies the computational one, but not vice versa.

3.1. *Commitment Schemes of Computational Secrecy*

The more standard commitment scheme is one in which the nonambiguity requirement is absolute (i.e., information theoretic) whereas the secrecy requirement is computational. For the sake of simplicity we refer to such schemes as commitment schemes.

Loosely speaking, a commitment scheme is an efficient *two-phase* two-party protocol through which one party, called the *sender*, can commit itself to a *value* so the following two conflicting requirements are satisfied:

1. *Secrecy*: At the end of the **commit phase**, the other party, called the **receiver**, does not gain any (computational) *knowledge* of the sender's value. This requirement has to be satisfied even if the receiver tries to cheat.
2. *Nonambiguity*: Given the transcript of the interaction in the commit phase, at most one value *exists* which the receiver may later (i.e., in the **reveal phase**) accept as a legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat.

In addition, it should be required that the protocol is **viable** in the sense that if both parties follow it, then, at the end of the second phase, the receiver gets the value committed to by the sender. Without loss of generality, the reveal phase may consist of merely letting the sender reveal the original value and the sequence of random coin tosses that it has used during the commit phase. The receiver will accept the value if and only if the supplied information matches its transcript of the interaction in the commit phase. The latter convention leads to the following definition (which refers explicitly only to the commit phase).

Definition 1 (Bit Commitment Scheme). A **bit commitment scheme** is a pair of probabilistic polynomial-time interactive machines, denoted (S, R) (for *sender* and *receiver*), satisfying:

- *Input specification*: The common input is an integer n presented in unary (*serving as the security parameter*). The private input to the sender is a bit, denoted v.
- *Secrecy*: The receiver (*even when deviating from the protocol in an arbitrary polynomial-time manner*) cannot distinguish a commitment to 0 from a commitment to 1. Namely, for every probabilistic polynomial-time machine R^* interacting with S, the random variables describing the output of R^* in the two cases, namely $(S(0), R^*)(1^n)$ and $(S(1), R^*)(1^n)$, are polynomially indistinguishable.
- *Nonambiguity*:
  Preliminaries:
  — A **receiver's view of an interaction** with the sender, denoted $(1^n, r, \bar{m})$, consists of the random coins used by the receiver (*denoted* r) and the sequence of messages received from the sender (*denoted* $\bar{m}$). (*In what follows, we sometimes omit* 1^n *from the receiver's view.*)
  — Let $\sigma \in \{0, 1\}$. We say that a receiver's view (*of such interaction*), $(1^n, r, \bar{m})$, is a **possible** σ**-commitment** if a string s exists such that $\bar{m}$ describes the messages received by R when R uses local coins r and interacts with machine S which uses local coins s and has input $(\sigma, 1^n)$.
  — We say that the receiver's view $(1^n, r, \bar{m})$ is **ambiguous** if it is both a possible 0-commitment and a possible 1-commitment.

  The **nonambiguity requirement** asserts that, for all but a negligible fraction of the coin tosses of the receiver, there is no sequence of messages (*of the sender*) which together with these coin tosses forms an ambiguous receiver view. Namely, for all but a negligible fraction of the $r \in \{0, 1\}^{\text{poly}(n)}$ there is no $\bar{m}$ such that $(1^n, r, \bar{m})$ is ambiguous.

The *secrecy requirement* (above) is a computational one; whereas the *nonambiguity requirement* has an information-theoretic flavor (i.e., it does not refer to computational powers). A dual definition, requiring information-theoretic secrecy and computational infeasibility of creating ambiguities, is presented in Section 3.2.

Naor showed that commitment schemes can be constructed using any pseudorandom generator [17], and the latter are known to exist provided that one-way functions exist [14], [13]. A much simpler commitment scheme can be constructed using any one-way permutation f. Using the results in [9], we may assume without loss of generality that the permutation f has a hard-core predicate, denoted b.

Construction 1 (Simple Bit Commitment). Let f: $\{0, 1\}^* \mapsto \{0, 1\}^*$ be a one-way permutation, and let b: $\{0, 1\}^* \mapsto \{0, 1\}$ be a hard-core predicate.

1. *Commit phase*: To commit to value $v \in \{0, 1\}$ (*using security parameter n*), the sender uniformly selects $s \in \{0, 1\}^n$ and sends the pair $(f(s), b(s) \oplus v)$ to the receiver.
2. *Reveal phase*: In the reveal phase the sender reveals the string s used in the commit phase. The receiver accepts the value v if $f(s) = \alpha$ and $b(s) \oplus v = \sigma$, where (α, σ) is the receiver's view of the commit phase.

The definition and the constructions of bit commitment schemes are easily extended to general commitment schemes enabling the sender to commit to a string rather than to a single bit. For the purposes of the current paper we need a commitment scheme by which one can commit to a ternary value. Extending the definition and the constructions to deal with this case is even more straightforward.

In the current paper we need commitment schemes with a seemingly stronger secrecy requirement than defined above. Specifically, instead of requiring secrecy with respect to all polynomial-time machines, we require secrecy with respect to all (not necessarily uniform) families of polynomial-size circuits. Assuming the existence of nonuniformly one-way functions (i.e., efficiently computable functions that cannot be inverted even by nonuniform families of polynomial-size circuits) commitment schemes with nonuniform secrecy can be constructed, following the same constructions used in the uniform case.

3.2. *Perfect Commitment Schemes*

The difference between commitment schemes (as defined in Section 3.1) and perfect commitment schemes (defined below) consists of a switching in scope of the secrecy and nonambiguity requirements. In commitment schemes (see Definition 1) the secrecy requirement is computational (i.e., refers only to probabilistic polynomial-time adversaries), whereas the nonambiguity requirement is information theoretic (and makes no reference to the computational power of the adversary). On the other hand, in perfect commitment schemes (see definition below) the secrecy requirement is information theoretic, whereas the nonambiguity requirement is computational (i.e., refers only to probabilistic polynomial-time adversaries). Hence, in some sense calling one of these schemes "perfect" is somewhat unfair to the other (yet, we do so in order to avoid cumbersome terms such as a "perfectly secret and computationally nonambiguous commitment scheme"). We remark that it is impossible to have a commitment scheme in which both the secrecy and nonambiguity requirements are information theoretic.

The Basic Definition

Loosely speaking, a perfect commitment scheme is an efficient *two-phase* two-party protocol through which the *sender* can commit itself to a *value* so the following two conflicting requirements are satisfied:

1. *Secrecy*: At the end of the *commit phase* the *receiver* does not gain any *information* of the sender's value.
2. *Nonambiguity*: It is *infeasible* for the sender to interact with the receiver so that the commit phase is successfully terminated and yet later it is *feasible* for the sender to perform the *reveal phase* in two different ways, leading the receiver to accept (as legal "openings") two different values.

Again, we require that the protocol is **viable** in the sense that if both parties follow it, then, at the end of the second phase, the receiver gets the value committed to by the sender. Using analogous conventions to the ones used in Section 3.1, we make the following definition.

Definition 2 (Perfect Bit Commitment Scheme). A **perfect bit commitment scheme** is a pair of probabilistic polynomial-time interactive machines, denoted (S, R) (for *sender* and *receiver*), satisfying:

- *Input specification*: As in Definition 1.
- *Secrecy*: For every probabilistic (*not necessarily polynomial-time*) machine R^* interacting with S, the random variables describing the output of R^* in the two cases, namely $(S(0), R^*)(1^n)$ and $(S(1), R^*)(1^n)$, are statistically close.
- *Nonambiguity*:
  Preliminaries. Fix any probabilistic polynomial-time algorithm F^*.
  - As in Definition 1, a **receiver's view of an interaction** with the sender, denoted $(1^n, r, \bar{m})$, consists of the random coins used by the receiver (r) and the sequence of messages received from the sender ($\bar{m}$). A **sender's view** of the same interaction, denoted $(1^n, s, \tilde{m})$, consists of the random coins used by the sender (s) and the sequence of messages received from the receiver ($\tilde{m}$). A **joint view** of the interaction is a pair consisting of corresponding receiver and sender views of the same interaction. (In what follows, we sometimes omit 1^n from the view.)
  - Let $\sigma \in \{0, 1\}$. We say that a joint view (of an interaction), $((r, \bar{m}), (s, \tilde{m}))$, has **a feasible σ-opening** (with respect to F^*) if, on input $(s, \tilde{m}, \bar{m}, \sigma)$, algorithm F^* outputs (*say, with probability at least* $1/2$) a string s' such that $\bar{m}$ describes the messages received by R when R uses local coins r and interacts with machine S which uses local coins s' and input $(\sigma, 1^n)$.
    (*Remark*: We stress that s' may, but need not, equal s. The output of algorithm F^* has to satisfy a relation which depends only on the receiver's view part of the input; the sender's view is supplied to algorithm F^* as additional help.)
  - We say that a joint view is **ambiguous** (with respect to F^*) if it has both a feasible 0-opening and a feasible 1-opening (with respect to F^*).

  The **nonambiguity requirement** asserts that, for all but a negligible fraction of the coin tosses of the receiver, it is infeasible for the sender to interact with the receiver so that the resulting joint view is ambiguous with respect to some probabilistic polynomial-time algorithm F^*. Namely, for every probabilistic polynomial-time interactive machine S^*, probabilistic polynomial-time algorithm F^*, polynomial

$p(\cdot)$, and all sufficiently large n, the probability that the joint view of the interaction between R and with S^*, on common input 1^n, is ambiguous with respect to F^*, is at most $1/p(n)$.

The nonambiguity requirement asserts that any efficient strategy S^* will fail to produce a joint view of interaction, which can later be (efficiently) opened in two different ways supporting two different values. As usual, events occurring with negligible probability are ignored. In the formulation of the nonambiguity requirement, S^* describes the (cheating) sender strategy in the commit phase, whereas F^* describes its strategy in the reveal phase. Hence, it is justified (and in fact necessary) to pass the sender's view of the interaction (between S^* and R) to algorithm F^*.

As in Definition 1, the secrecy requirement refers explicitly to the situation at the end of the commit phase, whereas the nonambiguity requirement implicitly assumes that the reveal phase takes the following form:

1. The committer sends to the receiver its initial private input, v, and the random coins, s, it has used in the commit phase.
2. The receiver verifies that v and s (together with the coins (r) used by R in the commit phase) indeed yield the messages R has received in the commit phase. Verification is done in polynomial time (by running the programs S and R).

Construction Based on Claw-Free Collections

Perfect commitment schemes can be constructed using a strong intractability assumption; specifically, the existence of claw-free collections (defined below). This assumption implies the existence of one-way functions, but it is not known whether the converse is true. Nevertheless, claw-free collections can be constructed under widely believed intractability assumptions such as the intractability of factoring and of DLP (see the Appendix). We start with a definition of claw-free collections. Loosely speaking, a claw-free collection consists of a set of pairs of functions which are easy to evaluate, both have the same range, and yet it is infeasible to find a range element together with preimages of it under each of these functions. We define claw-free collections in terms of the algorithms used to effect them; the index/function selection algorithm I, the domain-sampling algorithm D, and the function-evaluation algorithm F. Intuitively, algorithm I selects an *index*, i, which specifies a pair of domains, D_i^0 and D_i^1, and a pair of functions, f_i^0 and f_i^1, defined over the domains D_i^0 and D_i^1, respectively. On input σ and i, algorithm D selects randomly (but not necessarily uniformly) an element in the domain D_i^σ. On input σ, i and $x \in D_i^\sigma$, algorithm F computes the value of the function f_i^σ at x.

Definition 3 (Claw-Free Collection). A triple of algorithms, (I, D, F), is called a **claw-free collection** if the following conditions hold:

1. *The algorithms are efficient*: Both I and D are probabilistic polynomial time, whereas F is deterministic polynomial time. We denote by $f_i^\sigma(x)$ the output of F on input (σ, i, x), and by D_i^σ the support of the random variable $D(\sigma, i)$.

2. *Identical range distribution*: For every i in the range of algorithm I, the random variables $f_i^0(D(0,i))$ and $f_i^1(D(1,i))$ are identically distributed.
3. *Hard to form claws*: For every probabilistic polynomial time algorithm, A', every polynomial $p(\cdot)$, and all sufficiently large n's,

$$\mathrm{Prob}(f_{I_n}^0(X_n) = f_{I_n}^1(Y_n)) < \frac{1}{p(n)},$$

where I_n is a random variable describing the output distribution of algorithm I on input 1^n, and (X_n, Y_n) is a random variable describing the output of algorithm A' on input (random variable) I_n.

Item 2 in the definition requires that the functions f_i^0 and f_i^1 induce the same distribution when applied to elements selected at random by $D(0,i)$ and $D(1,i)$, respectively. A special case of interest is when both domains are identical (i.e., $D_i \overset{\text{def}}{=} D_i^0 = D_i^1$), the random variable $D(\sigma, i)$ is uniformly distributed over D_i, and the functions, f_i^0 and f_i^1, are permutations over D_i. Such a collection is called a collection of permutations. Anyhow, item 2 implies that many pairs (x, y) exist so that $f_i^0(x) = f_i^1(y)$ (e.g., in case of collections of permutations the number of such pairs is exactly $|D_i|$, but in general the number may be larger[6]). Such a pair is called a *claw*. The claw-forming adversary algorithm is given as input an index i, and tries to find a claw. It is required that although many claws do exist, efficient algorithms are unable to find claws. Clearly, a claw-free collection of permutations (resp. functions) yields a collection of strong one-way permutations (resp. functions). Examples of claw-free collections are presented in the Appendix. At this point we present a construction of perfect commitment schemes that uses a restricted type of a claw-free collection; specifically, we assume that the set of indices of the collection (i.e., the range of algorithm I) can be efficiently recognized (i.e., is in $\mathcal{BPP}$).

Construction 2 (Perfect Bit Commitment). Let (I, D, F) be a triplet of efficient algorithms.

1. *Commit phase*: To receive a commitment to a bit (*using security parameter n*), the receiver randomly generates an index i by invoking $I(1^n)$ and sends it to the sender. To commit to value $v \in \{0, 1\}$ (*upon receiving the message i from the receiver*), the sender checks if indeed i is in the range of $I(1^n)$, and if so the sender randomly generates a domain element s by invoking $D(v, i)$, computes $c \overset{\text{def}}{=} F(v, i, s)$, and sends c to the receiver. (*In case i is not in the range of $I(1^n)$ the sender aborts the protocol announcing that the receiver is cheating.*)
2. *Reveal phase*: In the reveal phase the sender reveals the string s used in the commit phase. The receiver accepts the value v if $F(v, i, s) = c$, where (i, c) is the receiver's (*partial*) view of the commit phase.

[6] In the general case, the number of claws equals $\sum_\alpha N_i^0(\alpha) \cdot N_i^1(\alpha)$, where $N_i^\sigma(\alpha) \overset{\text{def}}{=} |\{x\colon\ f_i^\sigma(x) = \alpha\}|$.

Proposition 1. *Let (I, D, F) be a claw-free collection* **with a probabilistic polynomial-time recognizable set of indices** *(i.e., the range of algorithm I is in $\mathcal{BPP}$). Then the protocol presented in Construction 2 constitutes a perfect bit commitment scheme.*

Proof. The secrecy requirement follows directly from property 2 of a claw-free collection (combined with the test $i \in I(1^n)$ conducted by the sender). The nonambiguity requirement follows from property 3 of a claw-free collection, using a standard reducibility argument. □

Remark 1. The definition of a claw-free collection may be relaxed in several ways maintaining the validity of Proposition 1. In particular, it suffices to require that the distributions, $f_i^0(D(0, i))$ and $f_i^1(D(1, i))$, are statistically close (rather than identical). Furthermore, this need not hold for all i's in the range of I, but rather for all $i \in I'$, where I' is an efficiently recognizable set so that $\text{Prob}(I(1^n) \notin I')$ is a negligible fraction.

Commitment Schemes with A Posteriori Secrecy

We conclude the discussion of perfect commitment schemes by introducing a relaxation of the secrecy requirement, that suffices for the purposes of the current work. The advantage in the relaxation is that it allows the construction of commitment schemes using any claw-free collection, thus waiving the additional requirement that the index set is efficiently recognizable.

Loosely speaking, we relax the secrecy requirement of perfect commitment schemes by requiring that it only holds whenever the receiver follows its prescribed program (denoted R). This seems strange since we do not really want to assume that the real receiver follows the prescribed program (but rather allow it to behave arbitrarily). The point is that a real receiver may disclose the coin tosses used by it in the commit phase at a later stage, say even after the reveal phase, and by doing so prove *a posteriori* that (at least in some weak sense) it was following the prescribed program. Actually, the receiver only proves that he behaved in a manner which is consistent with its program.

Definition 4 (Commitment Scheme with Perfect *A Posteriori* Secrecy). A **bit commitment scheme with perfect a posteriori secrecy** is defined as in Definition 3.2, except that the secrecy requirement is replaced by the following *a posteriori secrecy requirement*: for every string $r \in \{0, 1\}^{\text{poly}(n)}$ it holds that $(S(0), R_r)(1^n)$ and $(S(1), R_r)(1^n)$ are statistically close, where R_r denotes the execution of the interactive machine R when using internal coin tosses r.

The above *a posteriori secrecy requirement* can be further relaxed by requiring that it holds only for every $r \in R'$, where R' is efficiently recognizable and contains all but a negligible fraction of the strings of length $\text{poly}(n)$ (i.e., the number of coins used by the receiver on input 1^n). This relaxation of the *a posteriori secrecy requirement* is used for carrying out the proof of the following proposition using claw-free collections which have the identical range property only for most indices (see Remark 1 above).

Proposition 2. *Let (I, D, F) be a claw-free collection. Consider a modification of Construction* 2, *in which the sender's check, of whether i is in the range of $I(1^n)$, is omitted* (from the commit phase). *Then the resulting protocol constitutes a bit commitment scheme with perfect* a posteriori *secrecy.*

In contrast to Proposition 1, here the claw-free collection may not have an efficiently recognizable index set. Hence, the sender's check must be omitted from the commit phase. Yet, the receiver can later prove that the message sent by it during the commit phase (i.e., i) is indeed a valid index by disclosing the random coins it has used in order to generate i (using algorithm I). This will, *a posteriori*, convince the sender that its committed value was kept secret till the reveal phase. In case we used a claw-free collection of the relaxed sense discussed in Remark 1, the sender must also check that $i \in I'$. (Note that, for the purposes of the current paper, we could have further relaxed the definition of claw-free collections and settled for a set I', containing all but a negligible fraction of $I \cap \{0, 1\}^n$, such that I' has a constant-round interactive proof system.)

Proof. The *a posteriori* secrecy requirement follows directly from property 2 of a claw-free collection (combined with the assumption that i is indeed a valid index). The nonambiguity requirement follows as in Proposition 1. □

A typical application of a commitment scheme with perfect *a posteriori* secrecy is presented in the current work. In our setting the commitment scheme is used inside an interactive proof with the verifier playing the role of the sender (and the prover playing the role of the receiver). If the verifier *a posteriori* learns that the prover has been cheating, then the verifier rejects the input. Hence, no damage is caused, in this case, by the fact that the secrecy of the verifier's commitments might have been breached.

Extensions

As in the previous subsection, we need to extend the definitions and the constructions of perfect commitment schemes so that they enable the sender to commit to a string rather than to a single bit. The definitional extensions, omitted here, are quite straightforward. As for the constructions, we may use the following generalization of the commitment schemes presented above. In the commit phase the receiver generates and sends to the sender a single index i specifying a pair of functions (f_i^0, f_i^1). To commit to the bit string, $\bar{v} \stackrel{\text{def}}{=} \sigma_1 \cdots \sigma_m$, the sender sends to the receiver a sequence $(f_i^{\sigma_1}(s_1), \ldots, f_i^{\sigma_m}(s_m))$, where s_j is generated by invoking $D(\sigma_j, i)$. Preservation of perfect secrecy is argued by using the fact that the statistical difference between two product distributions is bounded by the sum of the componentwise statistical differences. Computational nonambiguity is argued using a standard reducibility argument while observing that two different "openings" of a commitment-sequence yield a claw in at least one component.

In addition, for the purposes of this paper, we need perfect commitment schemes with computational nonambiguity stated in nonuniform terms. Specifically, instead of requiring nonambiguity with respect to all polynomial-time machines, we require nonambiguity with respect to all (not necessarily uniform) families of polynomial-size circuits. Assuming the existence of claw-free collections for which even nonuniform polynomial-size circuits cannot form claws, perfect commitment schemes with nonuniform nonambiguity can be constructed. The constructions are identical to the ones used in the uniform case.

4. The Interactive Proof System

For the sake of clarity, we start by presenting a detailed description of the constant-round interactive proof, for Graph 3-Colorability (i.e., G3C), sketched in Section 2. This interactive proof employs two different commitment schemes. The first scheme is the simple commitment scheme (with "computational" secrecy) presented in Construction 1. We denote by $C_s(\sigma)$ the commitment of the sender, using coins s, to the (ternary) value σ. The second commitment scheme is a commitment scheme with perfect secrecy (see Section 3.2). For simplicity, we assume that the latter scheme has a commit phase in which the receiver sends one message to the sender who then replies with a single message (e.g., the schemes presented in Section 3.2). We denote by $P_{m,s}(\alpha)$ the commitment of the sender to string α, upon receiving message m (from the receiver) and when using coins s.

Construction 3 (A Round-Efficient Zero-Knowledge Proof for G3C).

- *Common input*: A simple (*3-colorable*) graph $G = (V, E)$. Let $n \overset{\text{def}}{=} |V|$, $t \overset{\text{def}}{=} 2n \cdot |E|$, and $V = \{1, \ldots, n\}$.
- *Auxiliary input to the prover*: A 3-coloring of G, denoted ψ.
- *Prover's preliminary step* (P0): The prover invokes the commit phase of the perfect commitment scheme, which results in sending to the verifier a message m.
- *Verifier's preliminary step* (V0): The verifier uniformly and independently selects a sequence of t edges, $\bar{E} \overset{\text{def}}{=} ((u_1, v_1), \ldots, (u_t, v_t)) \in E^t$, and sends to the prover a random commitment to these edges. Namely, the verifier uniformly selects $\bar{s} \in \{0, 1\}^{\text{poly}(n)}$ and sends $P_{m,\bar{s}}(\bar{E})$ to the prover.
- *Motivating remark*: At this point the verifier is committed to a sequence of t edges. (*This commitment is of perfect secrecy.*)
- *Prover's step* (P1): The prover uniformly and independently selects t permutations, $\pi_1, \ldots, \pi_t$, over $\{1, 2, 3\}$, and sets $\varphi_j(v) \overset{\text{def}}{=} \pi_j(\psi(v))$, for each $v \in V$ and $1 \leq j \leq t$. The prover uses the computational commitment scheme to commit itself to the colors of each of the vertices according to each 3-coloring. Namely, the prover uniformly and independently selects $s_{1,1}, \ldots, s_{n,t} \in \{0, 1\}^n$, computes $c_{i,j} = C_{s_{i,j}}(\varphi_j(i))$, for each $i \in V$ and $1 \leq j \leq t$, and sends $c_{1,1}, \ldots, c_{n,t}$ to the verifier.
- *Verifier's step* (V1): The verifier reveals the sequence $\bar{E} = ((u_1, v_1), \ldots, (u_t, v_t))$ to the prover. Namely, the verifier send $(\bar{s}, \bar{E})$ to the prover.
- *Motivating remark*: At this point the entire commitment of the verifier is revealed. The verifier now expects to receive, for each j, the colors assigned by the jth coloring to vertices u_j and v_j (the endpoints of the jth edge in $\bar{E}$).
- *Prover's step* (P2): The prover checks that the message just received from the verifier is indeed a valid revealing of the commitment made by the verifier at Step V0. Otherwise the prover halts immediately. We denote the sequence of t

edges, just revealed, by $(u_1, v_1), \ldots, (u_t, v_t)$. The prover uses the reveal phase of the computational commitment scheme in order to reveal (*to the verifier*), for each j, the jth coloring of vertices u_j and v_j. Namely, the prover sends to the verifier the sequence of fourtuples

$$(s_{u_1,1}, \varphi_1(u_1), s_{v_1,1}, \varphi_1(v_1)), \ldots, (s_{u_t,t}, \varphi_t(u_t), s_{v_t,t}, \varphi_t(v_t)).$$

- *Verifier's step* (V2): The verifier checks whether, for each j, the values in the jth fourtuple constitute a correct revealing of the commitments $c_{u_j,j}$ and $c_{v_j,j}$, and whether the corresponding values are different. Namely, upon receiving $(s_1, \sigma_1, s'_1, \tau_1)$ through $(s_t, \sigma_t, s'_t, \tau_t)$, the verifier checks whether, for each j, it holds that $c_{u_j,j} = C_{s_j}(\sigma_j)$, $c_{v_j,j} = C_{s'_j}(\tau_j)$, and $\sigma_j \neq \tau_j$ (*and both are in* $\{1, 2, 3\}$). If all conditions hold, then the verifier accepts. Otherwise it rejects.

We first assert that Construction 3 is indeed an interactive proof for G3C. Clearly, the verifier always accepts a common input in G3C. Suppose that the common input graph, $G = (V, E)$, is not in G3C. Clearly, each of the "committed colorings" sent by the prover in Step P1 contains at least one illegally colored edge. Using the perfect secrecy of the commitments sent by the verifier in Step V0, we deduce that at Step P1 the prover has "no idea" which edges the verifier asks to see (i.e., as far as the information available to the prover is concerned, each possibility is almost equally likely[7]). Hence, although the prover sends the "coloring commitment" after receiving the "edge commitment," the "edge commitment" is (almost) statistically independent of the "coloring commitment." It follows that the probability that all the "committed edges" have legally "committed coloring" is at most

$$\left(1 - \frac{1}{|E|} + \mu(|E|)\right)^t < e^{-n},$$

where μ is smaller than any nonnegligible function (and in particular $\mu(m) < 1/2m$). Hence, we get:

Proposition 3. *Construction* 3 *constitutes an interactive proof system for Graph* 3-*Colorability.*

5. The Simulator

We now turn to show that Construction 3 is indeed zero-knowledge (in the liberal sense allowing *expected* polynomial-time simulators). For every probabilistic (expected[8])

[7] The negligible difference in likelihood is due to the fact that the definition of perfect secrecy only requires the commitment distributions to be statistically close.

[8] Verifier strategies which run in expected polynomial time but not in strict polynomial time are considered for sake of elegance; see [11] and [10]. There are two common alternative definitions for the (expected) running time of an interactive machine; one alternative is to consider its executions with the prescribed counterpart (in our case the honest prover) and the other is to consider its executions with an arbitrary (i.e., worse-case) counterpart. Here we may use the more liberal alternative and consider all verifiers which run in expected polynomial time when the expectation is taken over the coin tosses of both the verifier and the honest prover.

polynomial-time interactive machine, V^*, we introduce an expected polynomial-time simulator, denoted M^*. The simulator starts by selecting and fixing a random tape, r, for V^*. Next, M^* simulates Step P0 by invoking the commit phase (of the perfect commitment scheme) and producing a message m. Given the input graph G, the random tape r, and the prover message m, the commitment message of the verifier V^* (for Step V0) is determined. Hence, M^* invokes V^*, on input G, random tape r, and message m, and gets the corresponding commitment message, denoted CM. The simulator proceeds in two steps.

S1. *Extracting the query edges*: The simulator M^* generates a sequence of $n \cdot t$ random commitments to dummy values (e.g., all values equal 1), and feeds it to V^*. (These commitments are via the regular commitment scheme and feeding them to V^* corresponds to the prover's Step P1.) In case V^* replies by revealing correctly a sequence of t edges, denoted $(u_1, v_1), \ldots, (u_t, v_t)$, the simulator records these edges and proceeds to the next step. In case the reply of V^* is not a valid revealing of the commitment message CM, the simulator halts outputting the current view of V^* (e.g., G, r, and the commitments to dummy values). Note that halting in such a case is consistent with the prover's behavior (in Step P2).

S2. *Generating an interaction that satisfies the query edges* (oversimplified exposition): Let $(u_1, v_1), \ldots, (u_t, v_t)$ denote the sequence of edges recorded in Step S1. The simulator M^* generates a sequence of $n \cdot t$ commitments, $c_{1,1}, \ldots, c_{n,t}$, so that, for each $j = 1, \ldots, t$, it holds that $c_{u_j,j}$ and $c_{v_j,j}$ are random commitments to two different random values in $\{1, 2, 3\}$ and all the other $c_{i,j}$'s are random commitments to dummy values (e.g., all values equal 1). The underlying values are called a *pseudocolorings*. The simulator feeds this sequence of commitments to V^* (which has been invoked from scratch with the same random-tape r and the same (Step P1)-message m). (Again, these commitments are via the regular commitment scheme and feeding them to V^* corresponds to the prover's Step P1.) If V^* replies by revealing correctly the *above-recorded* sequence of edges, then M^* can complete the simulation of a "real" interaction of V^* (by revealing the colors of the endpoints of these recorded edges). Otherwise, the entire Step S2 is repeated (until success occurs).

To illustrate the behavior of the simulator, assume that the program V^* always reveals correctly the commitment made in Step V0. In such a case the simulator will find out the query edges in Step S1, and using them in Step S2 it will simulate the interaction of V^* with the real prover. Using ideas as in [10], it can be shown that the simulation is computationally indistinguishable from the real interaction. Note that in this case (i.e., when V^* always replies properly), Step S2 of the simulator is performed only once.

Consider now a more complex case in which, on each possible sequence of internal coin tosses r, program V^* correctly reveals the commitment made in Step V0 only with probability 1/3. The probability in this statement is taken over all possible commitments generated to the dummy values (in the simulator Step S1). We first observe that the probability that V^* correctly reveals the commitment made in Step V0, *after receiving a random commitment to a sequence of pseudocolorings* (generated by the simulator in Step S2), is approximately 1/3. (Otherwise, we derive a contradiction to the computational secrecy of the commitment scheme used by the prover.) Hence, the simulator reaches Step S2 with probability $1/3$, and each execution of Step S2 is completed successfully with probability $p \approx 1/3$. It follows that the expected number of times that Step S2 is invoked when running the simulator is $(1/3) \cdot (1/p) \approx 1$.

We now consider the general case. Let $q(G, r)$ denote the probability that program V^*, on input graph G and random tape r, correctly reveals the commitment made in Step V0, *after receiving random commitments to dummy values* (generated in Step S1). Likewise, we denote by $p(G, r)$ the probability that V^* (on input graph G and random tape r) correctly reveals the commitment made in Step V0, *after receiving a random commitment to a sequence of pseudocolorings* (generated by the simulator in Step S2). As before the difference between $q(G, r)$ and $p(G, r)$ is negligible (in terms of the size of the graph G), otherwise a contradiction to the computational secrecy of the prover's commitment scheme is derived. We conclude that the simulator reaches Step S2 with probability $q \overset{\text{def}}{=} q(G, r)$, and each execution of Step S2 is completed successfully with probability $p \overset{\text{def}}{=} p(G, r)$. It follows that the expected number of times that Step S2 is invoked when running the simulator is q/p. Here is the bad news: we cannot guarantee that q/p is approximately 1 or even bounded by a polynomial in the input size (e.g., let $p = 2^{-n}$ and $q = 2^{-n/2}$, then the difference between them is negligible and yet q/p is not bounded by $\text{poly}(n)$). This is why the above description of the simulator is oversimplified and a modification is indeed required.

We make the simulator expected polynomial time by modifying Step S2 as follows. We first add an intermediate step, denoted S1.5, to be performed only if the simulator did not halt in Step S1. The purpose of Step S1.5 is to provide a good estimate of $q(G, r)$. The estimate is computed by repeating the experiment of Step S1 until a fixed (polynomial in $|G|$) number of correct[9] V^*-reveals are encountered (i.e., the estimate will be the ratio of the number of successes divided by the number of trials). We stress that, in case Step S1.5 is performed, the number of trials (in it) is not necessarily a polynomial but is rather $\text{poly}(|G|)/q(G, r)$, on the average. By fixing a sufficiently large polynomial, we can guarantee that with overwhelmingly high probability (i.e., $1 - 2^{-\text{poly}(|G|)}$) the estimate is within a constant factor of $q(G, r)$. Step S2 of the simulator is modified by adding a bound on the number of times it is performed, and if none of these executions yields a correct V^*-reveal, then the simulator outputs a *special symbol indicating* **time-out**. Specifically, Step S2 will be performed at most $\text{poly}(|G|)/\bar{q}$ times, where $\bar{q}$ is the estimate to $q(G, r)$ computed in Step S1.5. In addition, we modify the simulator so that if the verifier ever reveals a correct opening of the commitment that is different from the one recorded in Step S1, then the simulator halts outputting a *special symbol indicating* **ambiguity**. It can easily be verified that the modified simulator has an expected running time bounded by $q(G, r) \cdot \text{poly}(|G|)/q(G, r) = \text{poly}(|G|)$. Hence:

Claim 1. *The modified simulator runs in expected polynomial time.*

[9] We do not require here that the revealed string matches the one recorded in Step S1. The distinction, however, is immaterial in view of the last modification described below.

It is left to analyze the output distribution of the modified simulator. We start by reducing this analysis to the analysis of the output distribution of the original simulator. The modified simulator, hereafter denoted M^{**}, differs from the original one (i.e., M^*) in two types of executions in which M^{**} outputs special symbols, specifically "time-out" and "ambiguity," whereas the original simulator proceeds to the next iteration of Step S2. Hence, we need to bound the probability that these executions occur.

Claim 2. *The probability that the modified simulator outputs the time-out symbol is a negligible function of* $|G|$.

Proof. Let $\Delta(G, r)$ denote the probability that, on input a graph G and coin tosses r, the modified simulator outputs a special time-out symbol. Then

$$\begin{aligned}
\Delta(G,r) &= q(G,r)\cdot\sum_{i\geq 1}\mathrm{Prob}\left(\left\lfloor\frac{1}{\bar{q}}\right\rfloor = i\right)\cdot(1-p(G,r))^{i\cdot\mathrm{poly}(|G|)}\\
&< q(G,r)\cdot\left(\mathrm{Prob}\left(\frac{q(G,r)}{\bar{q}}=\Theta(1)\right)\cdot(1-p(G,r))^{\mathrm{poly}(|G|)/q(G,r)}\right.\\
&\qquad\left. +\ \mathrm{Prob}\left(\frac{q(G,r)}{\bar{q}}\neq\Theta(1)\right)\right)\\
&< q(G,r)\cdot(1-p(G,r))^{\mathrm{poly}(|G|)/q(G,r)}+2^{-|G|}.
\end{aligned}$$

In what follows, we ignore the additive $2^{-|G|}$ term. We now show that $\Delta(G, r)$ is a negligible function of $|G|$. Assume, to the contrary, that there are a polynomial $P(\cdot)$, an infinite sequence of graphs $\{G_n\}$ (with $|G_n| = n$), and an infinite sequence of random tapes $\{r_n\}$, such that $\Delta(G_n, r_n) > 1/P(n)$. It follows that for each such n we have $q(G_n, r_n) > 1/P(n)$. We consider two cases.

Case 1: *For infinitely many of these n's, it holds that* $p(G_n, r_n) \geq q(G_n, r_n)/2$. In such a case we get, for these n's,

$$\begin{aligned}
\Delta(G_n,r_n) &\leq (1-p(G_n,r_n))^{\mathrm{poly}(|G_n|)/q(G_n,r_n)}\\
&\leq \left(1-\frac{q(G_n,r_n)}{2}\right)^{\mathrm{poly}(|G_n|)/q(G_n,r_n)}\\
&< 2^{-\mathrm{poly}(|G_n|)/2},
\end{aligned}$$

which contradicts our hypothesis that $\Delta(G_n, r_n) > 1/\mathrm{poly}(n)$.

Case 2: *For infinitely many of these n's, it holds that* $p(G_n, r_n) < q(G_n, r_n)/2$. It follows that for these n's we have $|q(G_n, r_n) - p(G_n, r_n)| > P(n)/2$, which leads to contradiction of the computational secrecy of the commitment scheme (used by the prover).

Hence, contradiction follows in both cases. □

Claim 3. *The probability that the modified simulator outputs the* ambiguity *symbol is a negligible function of* $|G|$.

Proof. Intuitively, the claim follows by using the (computational) nonambiguity property of the verifier's commitment scheme. However, when trying to carry out the standard argument the following difficulty is encountered. The standard argument proceeds by contradiction and uses the machine V^*, invoked by the simulator, to do things assumed impossible (i.e., produce ambiguous commitments). The problem is that V^* might have revealed different values when invoked more than polynomially many times. Recall that the number of times Step S2 is performed is not bounded by a polynomial; only the expected number of times that Step S2 is performed (by the modified simulator) is bounded by a polynomial. Nevertheless, the problem is easily resolved by disregarding the executions of the modified simulator in which Step S2 is performed too many times. Specifically, assume by contraction that the "ambiguity" symbol is output with probability at least $1/P(|G|)$, for a polynomial $P(\cdot)$ and an infinite sequence of graphs. Then we can truncate the executions of M^{**} in which Step S2 is performed more than $2T(|G|) \cdot P(|G|)$ times, where $T(\cdot)$ denotes the expected running time of M^{**}. By an averaging argument it also follows that in these truncated executions M^{**} outputs an "ambiguity" symbol with nonnegligible probability (i.e., with probability at least $1/2P(|G|)$). Contradiction now follows using the standard techniques. □

To conclude, it suffices to show that the output of the *original* simulator (i.e., M^*) is computationally indistinguishable from the output of verifier V^* (when interacting with the prover).

Claim 4. *The ensemble* $\{M^*(G)\}_{G\in \mathrm{G3C}}$ *is computationally indistinguishable from the ensemble* $\{(P, V^*)(G)\}_{G\in \mathrm{G3C}}$, *where* $(P, V^*)(G)$ *denotes the output of* V^* *after an interaction with the prover on common input* G.

Proof. When trying to carry out the standard argument (i.e., as in [10]), we again encounter the difficulty mentioned in the proof of the previous claim. Namely, the standard argument proceeds by contradiction and uses the machine M^* to do things assumed impossible (i.e., distinguish computationally secure commitments to different values). However, here M^* is not strictly polynomial time, and furthermore M^* is not even guaranteed to be expected polynomial time. Yet, again, the problem is resolved by truncating the rare executions of M^* which are too long. Specifically, assume that the above ensembles are distinguished (by an efficient algorithm A) with gap $\varepsilon(G)$ (i.e., $\varepsilon(G) = |\mathrm{Prob}(A(M^*(G)) = 1) - \mathrm{Prob}(A((P, V^*)(G)) = 1)|$), and that $\varepsilon(G) \geq 1/P(|G|)$ for a polynomial $P(\cdot)$ and an infinite sequence of graphs $\{G_n\colon n \in S\}$ (with $|G_n| = n$). Defining a predicate R so that $R(y) = 1$ if y is an interaction-transcipt in which the verifier correctly reveals the commitment made in Step V0 and $R(y) = 0$ otherwise, we consider two cases:

Case 1: *For infinitely many* $n \in S$, *it holds that* $\mathrm{Prob}(R((P, V^*)(G_n)) = 1) \geq \varepsilon(G_n)/3$. On these G_n's, it is guaranteed that the expected number of times that Step S2 is performed (by M^*) is at most $3/\varepsilon(G_n) < 3P(|G_n|)$. Hence, runs of M^* in which Step S2 is repeated more than $T(|G_n|) \stackrel{\text{def}}{=} 6P(|G_n|)^2$ times occur with probability at most $1/2P(|G_n|)$. Thus, truncating the execution of M^* after $T(|G_n|)$ repetitions of Step S2 yields output that is at most $1/2P(|G_n|)$ away (in statistical distance) from the output of the original

M^*. It follows that algorithm A still distinguishes, with gap at least $\varepsilon(G_n) - 1/2P(|G_n|)$, the output of the truncated M^* from the real interaction with the prover. At this point, we may apply the standard techniques (see [10], but actually the proof here is simpler).

Case 2: *For infinitely many* $n \in S$, *it holds that* $\text{Prob}(R((P, V^*)(G_n)) = 1) < \varepsilon(G_n)/3$. It follows that, on these G_n's, with probability at least $1 - \varepsilon(G_n)/3$, the interaction of V^* with the real prover is suspended at Step V1. There are two subcases to consider:

- In the first subcase we assume that the simulator halts in Step S1 with probability at most $1 - \varepsilon(G_n)/2$. Thus, there is a gap, of at least $\varepsilon(G_n)/6$ between the probability that V^* correctly reveals its commitments when interacting with the prover and the probability that V^* correctly reveals its commitments when "interacting" with the simulator. In this case V^* is used to distinguish the commitments to dummy values (as produced by the simulator) from commitments to legal coloring (as produced by the prover), in contradiction to the computational secrecy of the prover's commitment scheme.
- In the second subcase we assume that the simulator halts in Step S1 with probability at least $1 - \varepsilon(G_n)/2$. This means that both the real and the simulated interactions are suspended with probability at least $1 - \varepsilon(G_n)/2$. Hence, algorithm A must distinguish such suspended interactions with gap at least $\varepsilon(G_n)/2$. It follows that algorithm A distinguishes commitments to dummy values (as appearing in suspended interactions produced by the simulator) from commitments to legal coloring (as appearing in suspended interactions with the prover).

Since in all cases we reached contradiction to the computational secrecy of the prover's commitment, the claim follows. □

Combining the above four claims, we get:

Proposition 4. *Construction* 3 *is zero-knowledge.*

6. Conclusion

Construction 3 can be modified so that weaker forms of perfect commitment schemes can be used. We refer specifically to commitment schemes with perfect *a posteriori* secrecy (see Section 3.2). In such schemes the secrecy is only established *a posteriori* by the receiver which discloses the coin tosses it has used in the commit phase. In our case the prover plays the role of the receiver, and the verifier plays the role of the sender. Hence, the prover may establish the secrecy of the verifier's commitment (of Step V0) by revealing, in Step P2, the coins it has used as receiver in Step P0. This suffices since, in case secrecy is not established, the verifier may reject. In such a case no harm has been caused since the secrecy of the perfect commitment scheme is used only to establish the soundness of the interactive proof. Hence, combining the above discussion with Propositions 2–4 we get:

Theorem 1. *If a claw-free collection exists, then every language in $\mathcal{NP}$ has a* **constant-round** *zero-knowledge interactive proof system.*

Acknowledgments

We would like to thank the anonymous referees for their useful comments.

Appendix. Examples of Claw-Free Collections

The following examples of claw-free collections have been discovered independently by many researchers. In particular, the DLP Claw-Free Collection has appeared in [3], and the Factoring Claw-Free Collection is an obvious modification of the construction appearing in [12].

A.1. The DLP Claw-Free Collection

We start by presenting a claw-free collection under the assumption that the Discrete Logarithm Problem (DLP) is intractable. Here we refer to the DLP for fields of prime cardinality. Namely, the input to DLP consists of a prime P, a primitive element of the multiplicative group mod P, and an element of the group. The problem is, given such a triplet (P, G, Y), to find an x such that $G^x \equiv Y \bmod P$. The *DLP intractability assumption* asserts that any efficient algorithm succeeds only with negligible probability (where the probability is taken over all possible inputs of specific length and the coin tosses of the algorithm).

Following is the description of the algorithms defining a collection of claw-free *permutations* (based on the above assumption). On input 1^n, the *index selection algorithm* I_{DLP} selects uniformly a prime, P, such that $2^{n-1} \leq P < 2^n$, a primitive element G in the multiplicative group modulo P, and an arbitrary member Z of that group, and outputs the index (P, G, Z). The domain of both functions with index (P, G, Z) is identical and equals the set $\{1, \ldots, P-1\}$. The *domain sampling algorithm*, D_{DLP}, uniformly selects an element of this set (i.e., $D_{\mathrm{DLP}}(\sigma, (P, G, Z))$ is uniformly distributed over $\{1, \ldots, P-1\}$, for both $\sigma \in \{0, 1\}$). As for the functions themselves, we set $F_{\mathrm{DLP}}(\sigma, (P, G, Z), x) \overset{\mathrm{def}}{=} Z^\sigma \cdot G^x \bmod P$, for both $\sigma \in \{0, 1\}$. The reader can easily verify that both functions are permutations over $\{1, \ldots, P-1\}$. Also, the ability to form a claw for the index (P, G, Z) yields the ability to find the discrete logarithm of Z mod P to base G (since $G^x \equiv Z \cdot G^y \bmod P$ yields $G^{x-y} \equiv Z \bmod P$). Hence, the ability to form claws for a nonnegligible fraction of the index set translates to a contradiction to the DLP intractability assumption.

The above collection *does not* have the additional property of having an efficiently recognizable index set, since it is not known how to recognize primitive elements modulo a prime efficiently. This can be amended by making a slightly stronger assumption concerning the intractability of DLP. Specifically, we assume that DLP is intractable even if one is given the factorization of the size of the multiplicative group (i.e., the factorization of $P-1$) as additional input. Such an assumption allows us to add the factorization of $P-1$ into the description of the index. This makes the index set efficiently recognizable (since P can first be tested for primality, as usual, and next test whether G is a primitive element by raising it to powers of the form $(P-1)/Q$ where Q is a prime factor of $P-1$). If DLP is hard also for primes of the form $2Q+1$, where Q is also a prime, life is even easier. To test whether G is a primitive element mod P just compute $G^2(\text{mod } P)$ and $G^{(P-1)/2}(\text{mod } P)$, and check whether either of them equals 1.

We remark that the above description assumes the existence of probabilistic polynomial-time algorithms for uniformly selecting primes and primitive elements. We only know of expected polynomial-time algorithms for these tasks. Furthermore, primality testers with no error are quite impractical, and therefore it is reasonable to use fast randomized algorithms (with negligible error probability) instead. Doing so we get something that is very close to a claw-free collection but does not quite achieve one (as with negligible probability the algorithms fail). We stress that this issue has no practical significance, yet if we wish to state a precise result, then the definition of claw-free collections needs to be slightly modified. Relaxing the definition of a claw-free collection requires a similar relaxation of the definition of perfect commitment schemes, so that Construction 2 remains valid. Details are omitted.

A.2. *The Factoring Claw-Free Collection*

We now show that a claw-free collection (of functions) does exist under the assumption that integer factorization is intractable for integers which are the product of two primes each congruent to 3 mod 4. Such composite numbers, hereafter referred to as *Blum integers*, have the property that the Jacobi symbol of -1 (relative to them) is 1 and half of the square roots of each quadratic residue, in the corresponding multiplicative group (modulo this composite), have Jacobi symbol 1. Let J_N^{+1} (resp. J_N^{-1}) denote the set of residues in the multiplicative group modulo N with Jacobi symbol $+1$ (resp. -1).

The *index-selecting algorithm*, denoted I_{FCT}, uniformly selects a Blum integer, by uniformly selecting two (n-bit) primes each congruent to 3 mod 4, and outputs their product, denoted N. The domains of the two functions with index N is J_N^{+1} and J_N^{-1}, respectively. The *domain-sampling algorithm*, D_{FCT}, on input σ and N, uniformly selects an element of $J_N^{(-1)^\sigma}$ (by uniformly selecting residues mod N and computing their Jacobi symbol). Finally, the functions themselves are defined by $F_{\text{FCT}}(\sigma, N, x) \overset{\text{def}}{=} f_N^\sigma(x) \overset{\text{def}}{=} x^2 \bmod N$, for both $\sigma \in \{0, 1\}$, where $x \in J_N^{(-1)^\sigma}$. Note that each of the two functions is 2-to-1.

The reader can easily verify that both $f_N^0(D(0, N))$ and $f_N^1(D(1, N))$ are uniformly distributed over the set of quadratic residues mod N. The difficulty of forming claws follows from the fact that a claw yields two residues, x and y, of different Jacobi symbol, (thus $x \neq \pm y$) such that $x^2 \equiv y^2 \pmod N$, and such residues yield a factorization of N.

References

[1] M. Blum, Coin Flipping by Phone, *Sigact News*, Vol. 15, No. 1, 1983.

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

[3] J. Boyar, M. Krentel, and S. Kurtz, A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs, *J. Cryptology*, Vol. 2, pp. 63–76, 1990.

[4] G. Brassard, D. Chaum, and C. Crépeau, Minimum Disclosure Proofs of Knowledge, *J. Comput. System Sci.*, Vol. 37, No. 2, pp. 156–189, 1988.

[5] G. Brassard, C. Crépeau, and M. Yung, Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols, *Theoret. Comput. Sci.*, Vol. 84, pp. 23–52, 1991.

[6] U. Feige and A. Shamir, Zero-Knowledge Proofs of Knowledge in Two Rounds, *Advances in Cryptology—Crypto* 89 (*Proceedings*), pp. 526–544, 1990.

[7] O. Goldreich, A Uniform-Complexity Treatment of Encryption and Zero-Knowledge, *J. Cryptology*, Vol. 6, pp. 21–53, 1993.

[8] O. Goldreich and H. Krawcyzk, On the Composition of Zero-Knowledge Proof Systems, *SIAM J. Comput.*, Vol. 25, No. 1, pp. 169–192, 1996.

[9] O. Goldreich and L. A. Levin, Hard-Core Predicates for Any One-Way Function, *Proc.* 21*st STOC*, pp. 25–32, 1989.

[10] O. Goldreich, S. Micali, and A. Wigderson, Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems, *J. Assoc. Comput. Mach.*, Vol. 38, No. 1, pp. 691–729, 1991.

[11] S. Goldwasser, S. Micali, and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems, *SIAM J. Comput.*, Vol. 18, No. 1, pp. 186–208, 1989.

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

[13] J. Hastad, Pseudorandom Generators Under Uniform Assumptions, *Proc.* 22*nd STOC*, pp. 395–404, 1990.

[14] R. Impagliazzo, L. A. Levin, and M. Luby, Pseudorandom Generation from One-Way Functions, *Proc.* 21*st STOC*, pp. 12–24, 1989.

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

[16] J. Kilian, On the Complexity of Bounded-Interaction and Noninteractive Zero-Knowledge Proofs, *Proc.* 35*th FOCS*, pp. 466–477, 1994.

[17] M. Naor, Bit Commitment Using Pseudorandomness, *J. Cryptology*, Vol. 4, pp. 151–158, 1991.