# Interactive and Zero-Knowledge Proofs

C. L. Groft

24 Sep 2001

## 1 Context and Definitions

Consider a general proof system for a language in **NP**. That is, suppose we have a language $L \in \mathbf{NP}$ and a string $x \in L$. For example, let $L$ be GRAPH-ISOMORPHISM and $x = (G_0, G_1)$ a pair of isomorphic graphs. Suppose Victor is not convinced that $x \in L$, so he calls his friend Peggy (who is infinitely powerful) and asks for a proof. Peggy agrees, because she's nice, applies her infinite power, and produces a witness $w(x)$ (in our example, $w(x)$ might be an isomorphism between $G_0$ and $G_1$). Victor then performs a verification procedure which takes less than $f(|x|)$ steps, where $f$ is a polynomial and $|x|$ is the length of $x$, and is convinced if and only if $w(x)$ is a valid witness for $x$. (That is, Victor checks the isomorphism to make sure it really is what Peggy claims it is.) If, however, Victor has $x' \notin L$, Peggy will not be able to falsely convince Victor that $x' \in L$. (There's no way to show an isomorphism between two nonisomorphic graphs.)

To be a bit more formal, a language $L$ is in **NP** if we can exhibit two machines, $P$ and $V$. $P$, or Peggy above, acts as a *prover*; she takes a string $x \in L$ as input, runs in finite but arbitrary time, and produces some string $w(x)$. $V$, or Victor, acts as a *verifier*; he takes $x$ and the output of $P$ as input, runs in time polynomial in $|x|$, and outputs "true" or "false". These machines must satisfy two properties:

1. *Completeness:* The proofs generated are always valid. That is, let $P(x)$ be the output of $P$ when run on $x$ and $V(x, w)$ be the output of $V$ when run on $x$ and $w$. If $x \in L$, we always have $V(x, P(x)) =$ "true".

2. *Soundness:* No valid proof exists for any nonmember of $L$. That is, let $x' \notin L$ and $P'$ be any machine which runs in finite time on $x'$. Then $V(x', P'(x')) =$ "false".

There are many languages that do not satisfy these criteria. For example, G-NONISO, the set of pairs of graphs $(x, y)$ where $x \ncong y$, is not thought to be in **NP**. It seems unreasonable to think that Peggy can convince Victor that two objects are similar, but cannot convince him that two objects are dissimilar. Can Peggy find a way to convince Victor that $x \ncong y$?

Since our current protocol is not adequate, let's expand it a bit, as detailed in [5]. For instance, we can change the one-way communication from Peggy to Victor to a dialogue, where Victor and Peggy exchange a bounded number of bounded messages. Also, we'll give Peggy and Victor random number generators, or RNGs, which are secure—that is, guaranteed to be uniformly random and secret from the other person. Finally, we'll relax the completeness and soundness rules a bit. We'll say a function $p(x)$ is *vanishingly small* if $p(x) \in o(|x|^{-d})$ for all $d > 0$. (For example, $p(x) = b^{-|x|}$ where $b > 1$ is vanishingly small.) The modified rules are as follows:

1. *Completeness:* If $x \in L$, then $V(x, P(x)) =$ "true" with probability $1 - p(x)$, where $p(x)$ is vanishingly small.

2. *Soundness:* If $x' \notin L$ and $P'$ is any machine terminating on $x'$, then $V(x', P'(x')) =$ "false" with probability $1 - q(x)$, where $q(x)$ is vanishingly small.

The class of languages which have a proof system under the above scheme will be called **IP** (for *interactive polynomial* time). Note that we know $\mathbf{NP} \subseteq \mathbf{IP}$ because any **NP** protocol can be seen as an **IP** protocol which ignores the extra features. It seems unlikely, however, that $\mathbf{IP} \subseteq \mathbf{NP}$, partly because there is an interactive protocol for GRAPH-NONISOMORPHISM, which is not thought to be in **NP**. Consider the following protocol:

1. Victor chooses $m = |x|$ ($x$ is the input string) random permutations $\pi_i \in_R S_n$ and $m$ random bits $b_i \in_R \{0, 1\}$. He then computes $m$ graphs $H_i = \pi_i G_{b_i}$ and sends the $H_i$ to Peggy.

2. Peggy, who is all-powerful, determines for each $i$ that $H_i$ is isomorphic to $G_{c_i}$, where $c_i \in \{0, 1\}$. She sends the $c_i$ to Victor.

3. Victor checks that $b_i = c_i$ for each $i$. He accepts iff this is the case.

Is this protocol correct? Let's check each condition, starting with completeness. If $G_0 \ncong G_1$, then Peggy can in fact determine $c_i$ for each $i$; if nothing else, she can perform a brute-force search of $G_0$'s equivalence class. She can then set $b_i = c_i$ for each $i$, at which point Victor will accept. Note that in this case Victor *always* accepts a valid instance of the language; such protocols are called *one-sided.* All of the protocols we discuss in this paper will be one-sided.

Now we must ask, is this protocol sound? Suppose $G_0 \cong G_1$. For any given $H_i$, Peggy cannot tell whether $H_i$ came from $G_0$ or $G_1$; in other words, she cannot determine $c_i$. The best she can do is guess the $b_i$s from a uniform random distribution; when she does so, the probability that $b_i = c_i$ for all $i$ is equal to $2^{-m}$, which is vanishingly small. This protocol is both sound and complete, and is therefore an interactive proof.

However, suppose Peggy is having problems with people abusing her infinite power. She doesn't mind people like Victor calling her up and asking for her help proving facts, but she doesn't want them gaining any additional knowledge from the process. She doesn't want Victor to be able to prove anything from the exchange that he couldn't prove just by assuming that $x \in L$. Also, she doesn't want Victor to go running to, say, Carol afterwards and be able to prove to her that $x \in L$. He can definitely do this with the current **NP** protocol, simply by handing Carol $w(x)$. However, we don't want Victor to be able to do this even if he cheats—that is, fails to follow the protocol. A protocol is *zero-knowledge* if this holds true. (Notice we're assuming that Victor isn't communicating with anyone else during the protocol itself. In other words, assume Victor has only one phone line and can't communicate any other way.)

To define this formally, consider what Victor "sees" during any protocol (not necessarily an interactive proof) for some input $x$. This includes his own random numbers and the information that Peggy give Victor (which may depend on Peggy's random numbers). If Victor is cheating, it may also include some information $H$, called the *history*, where $|H|$ is bounded by some polynomial of $|x|$. We'll use $U(x)$ to denote everything Victor sees on input $x$. If the protocol were deterministic, $U(x)$ would just be a string. Our protocol is nondeterministic, however, so $U(x)$ must be a *random variable* on the set of strings. The function $U$ can then be considered a family of random variables.

We can perform the same analysis on other randomized machines. Consider a general machine $M$, which has access to an RNG. $M(x)$ can also be considered a random variable, and $W_M : x \mapsto M(x)$ a family of random variables.

Finally, we can formally define zero-knowledge: A protocol between $P$ and $V$ is *zero-knowledge* if, for every choice of $V$, there exists a machine $M_V$ which runs in *expected* polynomial time on its input (which consists of $(x, H)$ where $x \in L$ and $|H|$ is bounded by a polynomial on $|x|$) such that $U$ and $W_{M_V}$ are indistinguishable. There are various definitions of "indistinguishable," some stronger than others, which are detailed in [5]. *Perfect zero-knowledge* refers to the case where the distributions are absolutely identical, while *statistical zero-knowledge* to the case where the difference between the probabilities in the two distributions of any given output is vanishingly small. Both of these interpretations assume an infinitely powerful judge. The most general form, *computational zero-knowledge* (or simply zero-knowledge), is also the most subtle. It states that, if $C_n$ is a sequence of Boolean circuits with a polynomially-bounded number of gates (or equivalently, a polynomial-time algorithm on a computer which outputs a bit), and $C(W(x))$ is the output of $C_{|W(x)|}$ when given $W(x)$ as input, then $|\Pr[C(U(x)) = 1] - \Pr[C(W_{M_V}(x)) = 1]|$ is vanishingly small. In other words, the judge is limited to polynomial runtime in this final case.

Heuristically, this means that if Victor simply assumed that $x \in L$, he could create any results himself that he could create by communicating with Peggy. Even more heuristically, this means that Victor gains no knowledge from the proof that $x \in L$ except the simple fact that $x \in L$. Note that all languages in **P** have trivial zero-knowledge proofs in which Peggy does nothing and Victor proves $x \in L$ for himself; thus the interesting cases will be where we do not know $L \in \mathbf{P}$.

# 2 Examples for "Isomorphism" Languages: G-ISO and QR

To illustrate the ideas above, consider this example from [6], a zero-knowledge interactive proof that two directed or undirected graphs $G_0$ and $G_1$ (each with, say, $n$ vertices) are isomorphic (assuming, of course, that they actually are isomorphic).

First, Peggy constructs $\phi \in S_n$[1], the isomorphism mapping $G_0$ to $G_1$. Then Peggy and Victor do the following for $m$ rounds, where $m$ is the length of the input:

1. Peggy chooses $\pi \in_R S_n$ ($\in_R$ denotes a random choice), computes $H = \pi G_1$, and sends $H$ to Victor.
2. Victor chooses and sends to Peggy $b \in_R \{0, 1\}$.
3. If $b = 1$, Peggy sends $\pi$ to Victor. If $b = 0$, Peggy sends $\pi \circ \phi$ to Victor. In each case, Peggy has sent an isomorphism between $G_b$ and $H$.
4. Victor checks that the function he received is in fact an isomorphism between $G_b$ and $H$. If this fails, Victor terminates (hangs up) and rejects the proof; otherwise, he proceeds.

If this procedure completes, Victor accepts the proof.

Is this proof valid? Let's look at each of our criteria.

- First, if $G_0 \cong G_1$, does Victor accept? It's fairly clear that he does.
- Second, can Peggy cheat? That is, if $G_0 \ncong G_1$, can Peggy convince Victor otherwise? Well, in each round, Peggy must send a graph $H$ to Victor. Since $G_0 \ncong G_1$, $H$ can be isomorphic to at most one of $G_0$ or $G_1$. Essentially, Peggy must guess what $b$ will be for this round. If she guesses correctly, all is well; she just sends the isomorphism between $H$ and $G_b$. If her guess is wrong, she's in trouble; she can't send an isomorphism between $H$ and $G_b$, because no isomorphism exists. Peggy has probability $1/2$ of guessing correctly on any round, so she has probability $1/2^m$ of falsely convincing Victor that $G_0 \cong G_1$. This probability is vanishingly small.
- Finally, can Victor cheat Peggy, by gaining more information about $G_0$ and $G_1$? It doesn't look like he can. All Victor sees are $m$ graphs, $m$ bits, and $m$ isomorphisms. No round does Victor any good by itself because $H$ is related to only one of the interesting graphs. Comparing different rounds probably will not do Victor any good either; for this to be helpful, $H$ would have to be the same in two rounds, which is not likely because

[1] $S_n$ is defined as the set of bijections from $N$ to $N$, where $N$ contains $n$ items. $N$ may be taken as the common set of vertices of $G_0$ and $G_1$.

Peggy chooses the $H$s.[2] So on a purely handwaving level, it looks like Victor gets nothing extra out of the proof. As it turns out, this protocol is perfectly zero-knowledge.

One thing to note about this algorithm is that constructing $\phi$ is the only expensive part of the algorithm for Peggy; once this is done (or if for some reason $\phi$ is already known), $P$ runs in polynomial time. In practice, this is very useful, since there aren't any infinitely powerful computers out there (at least, not that I have access to…). Also, one can reverse the abilities of Peggy and Victor—that is, make Victor infinitely powerful, and let Peggy be polynomially bounded—but hand Peggy the isomorphism $\phi$. In this case, Victor can tell that $G_0 \cong G_1$, since he's infinitely powerful, but he cannot tell which isomorphism Peggy has if there is more than one. This is called a *zero-knowledge proof of knowledge*, and it is also an interesting concept, but we will not return to it. Henceforth Victor will be mortal, and Peggy will be all-powerful.

Another very similar example from [5] is QR, the language of quadratic residues. Briefly, an integer $y$ is a residue mod $x$ if $y \in Z_x^*$ and, for some $r \in Z_x^*$, $r^2 \equiv y \pmod{x}$. If this holds true, $(x, y) \in$ QR. If $x$ is prime, it turns out to be easy not only to test membership, but to find $r$ for a given $y$. By the Chinese Remainder Theorem the problem is still easy provided one knows the prime factors of $x$. Otherwise the problem appears hard—factoring $x$ and using the Chinese Remainder Theorem is the fastest known solution.

This may not appear similar to the above problem at first. However, for a fixed $x$ consider the relation where $y \cong z$ iff $yz$ is a residue mod $x$. This can be shown to be an equivalence relation such that, if $y \cong z$, either neither of $y$ and $z$ are residues or both are. Thus $(x, y) \in$ QR iff $y \cong 1$ mod $x$. This suggests the following protocol: Using the prime factorization of $x$, Peggy computes $r$ where $r^2 = y$. Then Peggy and Victor do the following for $m = \log_2 x$ rounds:

1. Peggy chooses $s \in_R Z_x^*$, computes $z = s^2$, and sends $z$ to Victor.
2. Victor chooses and sends to Peggy $b \in_R \{0, 1\}$.
3. If $b = 0$, Peggy sends $s$ to Victor (proof that $1 \cong z$). If $b = 1$, Peggy sends $rs$ to Victor (proof that $y \cong z$).
4. Victor checks that $t^2 = z$ or $t^2 = yz$, where $t$ is the number he received, depending on $b$. If this fails, Victor terminates and rejects; otherwise, he proceeds.

If this procedure completes, Victor accepts the proof.

The justification for this protocol is exactly as above. Victor accepts if $y$ is a residue, accepts with probability $1/2^m$ if it is not (Peggy must guess whether $z$ or $yz$ must be a quadratic residue, and will be right half the time), and can gain no extra knowledge from the exchange. This protocol is also perfectly zero-knowledge.

[2] Eventually Peggy will have to repeat, but Victor will have to keep a superpolynomial amount of information to detect that fact. In any case, since there are so few rounds, Peggy can watch for this case herself and choose a new graph should she repeat one.

# 3 Examples for "Nonisomorphism" Languages

We can also look at the negated languages of the above; that is, the pairs of graphs $(G_0, G_1)$ where $G_0 \ncong G_1$ and the pairs $(x, y)$ where $y$ is not a quadratic residue mod $x$. We'll start with non-isomorphism (also covered in [6]): two graphs $G_0$ and $G_1$, both with $n$ vertices and the same number of edges, but with $G_0 \ncong G_1$. Can Peggy prove this fact to Victor without giving away anything else?

Let's consider the protocol from earlier. It fulfills the completeness and soundness conditions, just as before. However, it is not zero-knowledge. Suppose Victor knows a graph $K$ such that either $K \cong G_0$ or $K \cong G_1$, but he does not know which is true. (How he knows this is not important; perhaps it came from some trusted source.) He can substitute $K$ for one of the $H_i$. When Peggy's results are in, he will know whether $K \cong G_0$ or $K \cong G_1$ while still being reasonably certain that $G_0 \ncong G_1$. Victor cannot be allowed to gain this knowledge!

To make this protocol zero-knowledge, Victor essentially must prove to Peggy that he knows for each $H_i$ whether $H_i \cong G_0$ or $H_i \cong G_1$ (so that Victor can't cheat) without revealing which is true (which would allow Peggy to cheat). To do this, we'll start with the above protocol but add the following subprotocol, to be followed for each $H_i$ either after sending each $H_i$ or after sending all of the $H_i$:

1. For each $j \in \{1, \ldots, m\}$, Victor chooses $r_{0j}, r_{1j} \in_R S_n$, computes $K_{kj} = r_{kj} G_k$, and sends either $(K_{0j}, K_{1j})$ or $(K_{1j}, K_{0j})$ (chosen randomly) to Peggy.

2. Peggy sends $v_1, \ldots, v_m \in_R \{0, 1\}$ to Victor.

3. For each $j$, if $v_j = 0$, Victor sends $(r_{0j}, r_{1j})$ (thus proving that the $K_{kj}$ really came from the $G_k$). If $v_j = 1$, Victor sends $\pi_i \circ r_{b_{ij}}^{-1}$, an isomorphism between $K_{b_{ij}}$ and $H_i$ (thus proving that the pairs are connected to the interesting graph).

4. Peggy checks Victor's information; if it is valid, she continues with the main protocol.

Suppose Victor tries to cheat with one of the $H_i$. Then to trick Peggy, he must guess whether Peggy will ask about $(G_0, G_1)$ or $H_i$ for each pair of graphs $\{K_{0j}, K_{1j}\}$. He can do this with probability $1/2$ for each round, or probability $1/2^m$ for the entire subprotocol. As before, this is low enough.

Can Peggy cheat with this new information—that is, figure out $b_i$ from this dialogue? It doesn't appear possible. All she has are $m$ unordered pairs of graphs, and an equivalence between each pair and either $(G_0, G_1)$ or $H_i$, never both. Unless a graph appears twice, which is vanishingly unlikely (and which Victor can avoid anyway), Peggy can't establish a connection between $(G_0, G_1)$ and $H_i$. (Note that randomly ordering each pair in step 1 above is essential;

if the ordering is fixed, Peggy can send $v_j = 1$ for some $j$ and cheat with the result.)

As noted above, G-NONISO is thought not to be in **NP**.

As a side note (which may be safely ignored if it sounds like gibberish), if one chooses to send all of the $H_i$ at once, then follow the subprotocol in parallel for all the graphs, the protocol completes with Victor and Peggy each sending only two messages. The same is true for the QNR protocol described below. This is sometimes expressed as G-NONISO, QNR $\in \mathbf{IP}_2$. It turns out that using this fact, one can show that $\Sigma_2^p = \Pi_2^p$ if G-ISO $\in$ **NPC**. Since it is widely believed that $\Sigma_2^p \neq \Pi_2^p$, it is probably true that G-ISO $\notin$ **NPC**.

As one might expect, we can negate QR the same way we negated G-ISO. Define QNR (as in [5]) as the set of pairs $(x, y)$ where $y$ is a nonresidue mod $x$ and has Jacobi symbol 1. (For those who are curious, the Jacobi symbol is a number-theoretic function of $x$ and $y$ which (a) can be efficiently calculated even without knowing the factorization of $x$, and (b) is always equal to 1 if $y$ is a residue mod $x$. For details, consult the nearest textbook on number theory.) Using the isomorphism given in part 2, we are now trying to show that $1 \not\cong y$. Unlike G-NONISO, QNR is known to be in **NP** (a factorization of $x$ will serve as a witness).

The protocol to prove QNR in zero-knowledge is almost identical to that for G-NONISO. Peggy first factors $x$ (and, if she wishes, checks that $y$ is in fact a nonresidue). Peggy and Victor then complete the following for $m$ rounds:

1. Victor chooses a random integer $r \in_R Z_x^*$ and a random bit $b \in_R \{0, 1\}$. If $b = 0$, Victor computes $a = r^2$; if $b = 1$, Victor computes $a = yr^2$. (Thus $a$ is a nonresidue iff $b = 1$.) He sends $a$ to Peggy.

2. For each $j \in \{1, \ldots, m\}$, Victor chooses $s_{0j}, s_{1j} \in_R Z_x^*$, computes $d_{0j} = s_{0j}^2$ and $d_{1j} = ys_{1j}^2$, and sends the randomly ordered pairs $\{d_{0j}, d_{1j}\}$ to Peggy.

3. Peggy sends $v_1, \ldots, v_m \in_R \{0, 1\}$ to Victor.

4. For each $j$, if $v_j = 0$, Victor sends $(s_{0j}, s_{1j})$ (thus proving that the $d_{kj}$ are connected to 1 and $y$). If $v_j = 1$, Victor computes either $f_j = rs_{0j}$ (if $b = 0$) or $f_j = yrs_{1j}$ (if $b = 1$) and sends $f_j$ to Peggy (thus proving that $a$ is connected to the $d_{kj}$).

5. Peggy checks Victor's information. If $v_j = 0$, Peggy checks that $s_{0j}^2 = d_{0j}$ and $ys_{1j}^2 = d_{1j}$. If $v_j = 1$, Peggy finds $f_j^2$. If $b = 0$, she will find $f_j^2 = r^2 s_{0j}^2 = ad_{0j}$; if $b = 1$, she will find $f_j^2 = yr^2 ys_{1j}^2 = ad_{1j}$. Assuming this is true for all $j$, Peggy determines whether $a$ is a nonresidue mod $x$. If it is, Peggy sends $c = 1$; otherwise she sends $c = 0$.

6. Victor checks that $b = c$. He halts and rejects if this is not the case.

Victor accepts if the protocol completes.

Again, the validity argument is a copy of a previous argument. The protocol completes if $y$ is a nonresidue and nobody cheats. Peggy cannot cheat, because if $y$ is a residue, $a$ will always be a residue. Peggy can get no information from the subprotocol, because a relation between $a$ and $y$ or between $a$ and 1 is never shown. Hence Peggy has probability $1/2$ of succeding on each round and probability $1/2^m$ of fooling Victor completely. Also, if Victor tries to cheat on one round, he only succeeds with probability $1/2^m$.

Though it is more difficult to prove, the above two protocols are also perfectly zero-knowledge.

# 4 Zero-Knowledge on an NP-complete Problem

Let the set of languages with zero-knowledge interactive proofs be denoted **ZP**. Clearly we have $\mathbf{ZP} \subseteq \mathbf{IP}$. We will eventually prove that, given certain assumptions, $ZK = IP$. An important first step is proving that $\mathbf{NP} \subseteq \mathbf{ZP}$; that is, given a language $L \in \mathbf{NP}$, there is a zero-knowledge proof for $L$.

We will show this, as [6] does, by looking at the NP-complete problems. Specifically, we will look at 3-COLOR, the set of graphs which have 3-colorings. (Remember that a 3-coloring of a graph $G = (V, E)$ is a function $\phi : V \to \{1, 2, 3\}$ where $(u, v) \in E$ implies $\phi(u) \neq \phi(v)$.) The assumption we will make here is that effective bit-commitment (or color-commitment) schemes exist. Formally, a *color-commitment* scheme is a function $f$, computable in polynomial time, which takes a color and a string called a key as input and outputs a string. $f$ is such that $f(c, k) \neq f(d, l)$ for all $k$ and $l$ if $c \neq d$. Also, given only $f(c, k_i)$ where $i \in \{1, \ldots, m\}$ and $|k_i| = m$, it should not be feasible to compute $c$. As we will see, such a scheme allows Peggy to commit to a color for a particular vertex without revealing that color until later, if at all.

Given the existence of such a function $f$, we can construct a protocol as in [6]. Peggy has $G = (V, E)$, where we'll assume $V = \{1, \ldots, n\}$, $|E| = m$, and $G$ is connected; the first thing she does is construct a 3-coloring on $G$, which we'll call $\phi$. The following then proceeds for $m^2$ rounds:

1. Peggy chooses $\pi \in_R S_3$; then $\phi' = \pi \circ \phi$ is also a 3-coloring of $G$. Then, for $i \in \{1, \ldots, n\}$, Peggy chooses $k_i \in_R \{0, 1\}^n$ (random $n$-bit strings). She calculates $F_i = f(\phi'(i), k_i)$ (the color-commitments) and sends the $F_i$ to Victor.

2. Victor chooses $e = (u, v) \in_R E$ and sends $e$ to Peggy.

3. Peggy verifies that $e \in E$ and sends $\phi'(u)$, $\phi'(v)$, $k_u$, and $k_v$ to Victor.

4. Victor verifies that $F_u = f(\phi'(u), k_u)$, $F_v = f(\phi'(v), k_v)$, and $\phi'(u) \neq \phi'(v)$.

The protocol works if $G$ has a 3-coloring and Peggy and Victor are both honest, since by definition of 3-coloring $\phi'(u) \neq \phi'(v)$ whenever $(u, v) \in E$. It doesn't look like Victor can cheat; all he sends is an edge of $G$ (if he sends anything else, he'll be caught) and all he sees are the colors of $u$ and $v$ (the other color-commitments might as well be garbage for all Victor can make of them). Furthermore, since $\phi'$ is chosen randomly for each round, Victor can't even tell what $\phi(u)$ and $\phi(v)$ are; all he knows is that they're not the same, which is inherent in the definition of a 3-coloring.

Can Peggy cheat? Suppose $G$ is not 3-colorable; then no matter what coloring Peggy commits to in step 1, that coloring will contain at least one out of $m$ bad edges, or edges whose endpoints have the same color.[3] If Victor chooses a bad edge, Peggy is out of luck. The probability that Peggy will fool Victor in any round is therefore at most $1-\frac{1}{m}$, and the probability that Victor will falsely accept $G$ is at most $(1-\frac{1}{m})^{m^2}$. This probability asymptotically approaches $e^{-m}$, which is vanishingly small.

Thus we know that 3-COLOR has a zero-knowledge proof. Since any language $L \in \mathbf{NP}$ can be reduced to 3-COLOR, $L$ has a zero-knowledge proof system as well: given $x \in L$, transform $x$ into an instance $t(x) \in$ 3-COLOR and perform a zero-knowledge proof on $t(x)$.

Note that, if 3-COLOR is a hard problem, the above protocol is only computationally zero-knowledge, not perfectly or statistically zero-knowledge. A simulator cannot hope to generate a real 3-coloring, and an infinitely powerful judge could break the color-commitments in a transcript, see that the "3-coloring" to which the simulator commited was not in fact a 3-coloring, and reliably pronounce the transcript a forgery.

# 5 RSR Languages

Given the similarity of the G-ISO and QR protocols, as well as that of the G-NONISO and QNR protocols, one might wonder if there exists a class of languages, all of which have similar zero-knowledge proofs. As it turns out, the *random self-reducible languages*, or RSR languages, form such a class. The concept of RSR was introduced by Tompa and Woll, who proved that all RSR languages (and their complements, called the co-RSR languages) have perfect ZK proofs.

We will define these languages as they are defined in [3]. Suppose $N$ is a countably infinite set, and for each $x \in N$, $R_x$ is a binary relation (though probably not a equivalence relation) which can be verified efficiently. Let the domain of $R_x$ be the set of all $z$ where $z\, R_x\, w$ for some $w$. We call $w$ an $x$-witness for $z$ when this holds. We may then define a language $L_R = \{(x, z) : \exists w \text{ s.t. } z\, R_x\, w\}$; doing so, we have $L_R \in NP$.

We want to have two efficient algorithms, $S$ and $G$, and two additional algorithms $A_1$ and $A_2$ related to $S$. We call $S$ a sampling algorithm; it takes inputs $(x, z) \in L_R$ and $r$ (where $r$ is ideally a randomly chosen series of bits) and outputs $y$ where $(x, y) \in L_R$ and $y$ is uniformly distributed over the domain of

[3] It is possible that Peggy will send an $F_i$ in step 1 that is not a color-commitment. In this case, any edge with $i$ as an endpoint can be considered a bad edge. Since the graph is connected, there is at least one such edge, so the argument still applies.

$R_x$. Given $x$, $y$, $r$ (the same bits as above) and a witness for $(x, y)$, $A_1$ should be able to output a witness for $(x, z)$; $A_2$ should output a witness for $(x, y)$ given $x$, $z$, $r$, and a witness for $(x, z)$. $S$ is necessarily an garbage-in garbage-out algorithm; if $(x, z) \notin L_R$, then even if $S$ terminates on its input, $(x, y) \notin L_R$. $G$, a generating algorithm, takes $x$ as input and outputs $(z, w) \in R_x$, where $z$ is uniformly distributed over the domain of $R_x$ and $w$ is a randomly chosen $x$-witness for $z$.

If the above algorithms exist, $L_R$ is an RSR language. The following (for $|x|$ rounds) is a zero-knowledge proof by Peggy to Victor that $(x, z) \in L_R$, where $(z, w) \in R_x$:

1. Peggy randomly chooses a bitstream $r$, calculates $y = S(x, z, r)$, and sends $y$ to Victor.
2. Victor chooses $b \in_R \{0, 1\}$ and sends $b$ to Peggy.
3. If $b = 1$, Peggy sends $r$ to Victor; if $b = 0$, she calculates $v = A_2(x, z, r, w)$ ($v$ is a witness that $(x, y) \in L_R$) and sends $v$ to Victor.
4. Victor checks that $y = S(x, z, r)$ if $b = 1$, or that $(y, v) \in R_x$ if $b = 0$.

The above has been a bit abstract and dense, so let's given an example—graph isomorphism. In this case, $x$ is a graph, $R_x$ is the set of graphs $z$ isomorphic to $x$ paired with the isomorphisms $\pi$ (that is, $z = \pi x$), and $L_R$ is the set of pairs of isomorphic graphs. A round of the protocol proving that $x \simeq z$ translates as follows:

1. Peggy randomly chooses a permutation $\phi$, calculates $y = \phi z$, and sends $y$ to Victor.
2. Victor generates and sends a random bit $b$.
3. If $b = 1$, Peggy sends $\phi$ to Victor; if $b = 0$, Peggy sends $\phi \circ \pi$.
4. Victor checks that $y = \phi z$ or $y = (\phi \circ \pi)x$.

This is simply the usual protocol for graph-isomorphism, as first given in [5]. Other examples include quadratic-residuosity ($x$ is an integer and $z = w^2$ in $Z_x$) and discrete-logarithm ($x = (p, g)$ where $p$ is an integer and $g \in Z_p^*$, $z = g^w$). Also, the complementary languages have zero-knowledge proofs similar to those of graph-nonisomorphism. In each round, Victor generates an object either from $S$ or from $G$; after Victor proves to Peggy that he knows where the object came from without revealing its origin, Peggy applies her infinite power, reasons out the object's origin, and tells Victor.

One result given in [3] states that the language of monotone formulae—that is, boolean formulae consisting only of ANDs and ORs—over RSR languages and the language of monotone formulae over co-RSR languages both have perfect zero-knowledge proofs. The languages are not quite RSR or co-RSR, but they are close enough that they can be treated as essentially interchangable.

Another interesting proof presented therein is that of $S \vee T$, where $S$ is a monotone formula over co-RSR statements and $T$ is a monotone formula over RSR statements. For example, let $S = (G_0 \not\simeq G_1)$ and $T = (H_0 \simeq H_1)$. A perfect ZK proof that this statement is true would proceed as follows, for $m$ rounds:

1. Victor chooses a permutation $\psi \in_R S_n$ and a bit $b \in_R \{0, 1\}$ and calculates $G = \psi G_b$. He sends $G$ to Peggy.
2. If Peggy can show that $G \not\simeq G_0$, she generates a graph $H = \phi H_1$. Otherwise she generates $H = \phi H_0$. Either way, she sends $H$ to Victor.
3. Victor sends $\psi$ to Peggy (simultaneously revealing $b$).
4. Peggy sends an isomorphism between $H$ and $H_b$.

Let's see how this works. Suppose $G_0 \not\simeq G_1$. Immediately after seeing $G$, Peggy will be able to calculate $b$ and generate $H \simeq H_b$. Thus, when Victor sends $\psi$, Peggy will be ready with the isomorphism $\phi$. If $H_0 = \chi H_1$ for some $\chi$, then Peggy does not need to know $b$ before finding $H$. She can assume $b = 0$; if she is wrong, she can send $\phi \circ \chi$ instead of $\phi$.

A similar protocol will work for any formula over co-RSR languages paired with any formula over RSR languages.

# 6 ZK Proofs for Generator and Non-Generator

Let $g$ be an element of $Z_n^*$, and let $Y(n, g, x)$ be a zero-knowledge protocol which proves that $x$ is a power of $g$ mod $n$. This protocol does not have to be the same as that given above. In fact, a protocol is given in [4] which uses the idea of "smooth numbers" to achieve much lower communication complexity. The important point for this protocol is that *some* protocol exists.

Consider the language GEN $= \{(n, g) : \langle g \rangle = Z_n^*\}$, where $\langle g \rangle$ is the set of powers of $g$ in $Z_n^*$; also known as the cyclic subgroup of $Z_n^*$ generated by $g$. The following, for $|n|$ rounds, is a zero-knowledge proof that $(g, n) \in$ GEN.

1. Victor chooses an element $x \in_R Z_n^*$ and sends $x$ to Peggy.
2. Peggy and Victor perform the zero-knowledge protocol $Y(n, g, x)$. That is, Peggy proves that $x$ is a power of $g$.

Victor halts and rejects if any step of the above rejects; he accepts if the protocol completes.

If $g$ generates $Z_n^*$, this protocol succeeds. Lagrange's Theorem states that the size of $\langle g \rangle$ divides the size of $Z_n^*$, so if $g$ does not generate $Z_n^*$, a randomly chosen $x \in Z_n^*$ will be outside $\langle g \rangle$ with at least probability 1/2; after $|n|$ rounds, the probability of success is vanishingly small.

The question of zero-knowledge is a bit more subtle. There is the possibility that Victor could insert an interesting element $x' \in Z_n^*$ at some step of the protocol, and we have made no provisions for catching him at this. Actually, there is no need to do so. The only thing Victor learns from the round in which he cheats is that $x' \in \langle g \rangle$. After the protocol terminates, he knows that $\langle g \rangle = Z_n^*$, which implies that $x' \in \langle g \rangle$ anyway. Thus, Victor gains nothing by cheating.

Note, however, that the above no longer holds true if, in fact, $g$ does *not* generate $Z_n^*$ and Peggy is trying to convince Victor otherwise. If Victor knows or guesses that Peggy is doing so, he can insert several interesting elements $x_1, x_2, \ldots$ into the protocol; if Peggy finally gives up on $x_n$, Victor knows that $x_1, \ldots, x_{n-1}$ are powers of $g$ and that $x_n$ probably is not a power of $g$ (else why would Peggy stop?) This is an interesting feature which I have not seen in any other ZK protocols. The feature, which I will dub *nonabusability* for lack of a better term, appears to be good for Victor because it helps keep Peggy honest.

Also, there is a ZK protocol for NONGEN $= \{(n, g) : \langle g \rangle \neq Z_n^*\}$. Simply put, Peggy chooses an element $x \in_R Z_n^*$ which is not a power of $g$ and proves this fact to Victor in zero-knowledge. This protocol is complete, sound, and zero-knowledge for the same reasons that the subprotocol is complete, sound, and zero-knowledge (whatever those reasons are), and the subprotocol's existence is guaranteed by the discussion above on RSR languages. The one subtlety to this proof is that, if the discrete-log problem is difficult, it may not be possible for a simulator to create a non-power of $g$ to begin the protocol. However, if the problem is indeed difficult, it is probably the case that no family of circuits will be able to reliably detect that the simulated protocol begins with a non-power of $g$. Thus the protocol will be computationally zero-knowledge, but will not necessarily be perfectly zero-knowledge.

As far as I know, the results of this section are original; if anyone knows otherwise, please let me know.

# 7 How Weak Is the Bit-Commitment Assumption?

Currently our only proof that all $NP$ languages have ZK protocols is given by [6], and this proof assumes that secure bit-commitment schemes exist. [6] notes, and we will see later, that one can prove **ZK** = **IP** under the same assumption. While the assumption is something of a blemish—we'd prefer not to need it—it may not be a very strong assumption. [2] shows that given any RSR language not in $P$, one can construct a bit-commitment scheme. This scheme will be slightly different from that discussed previously, which was impossible to forge and infeasible to break; this scheme will be impossible to break and infeasible to forge.

Thus, if bit-commitment schemes do not exist, it is probably the case that all RSR languages (and by extension, all co-RSR languages) are easy. In this case, not only does every problem we have discussed besides the NP-complete problems become trivial, but Bad Things Happen to various complexity classes as well. More positively, we know that protocols which rely on the bit-commitment assumption are almost certainly strong enough [2].

In this case, a bit-commitment scheme is a protocol between Alice and Bob, two polynomial-time machines. In the first part, Alice commits to a bit $b$ by generating a random string $r$, computing some function $BC(b, r)$, and sending the result to Bob. Bob should not be able to predict $b$ from $BC(b, r)$. In the second part, Alice reveals her commitment to Bob by revealing $b$ and $r$. Alice should not be able to find $r, r'$ where $BC(0, r) = BC(1, r')$.

The following protocol will use graph-isomorphism as its interesting problem, but any RSR language can be substituted. To set up the protocol, Bob generates two isomorphic graphs $G_0$ and $G_1 = \psi G_0$, where $\psi$ is chosen randomly. He then gives them to Alice and proves in zero-knowledge that the graphs are isomorphic.[4]

To commit to a bit $b$, Alice generates a simulated transcript of a ZK proof with an honest verifier that $G_0 \simeq G_1$. (The definition of zero-knowledge guarantees that she can do this.) In this case, the transcript consists of $m$ bits $b_1, \ldots, b_m$, $m$ randomly chosen permutations $\phi_1, \ldots, \phi_m$, and $m$ graphs $H_i = \phi_i G_{b_i}$. Alice then calculates the bits $c_i = b \oplus b_i$ and sends the $H_i$ and $c_i$ to Bob. Since Bob cannot find $b_i$ for any $i$, he cannot calculate $b$.

To open a commitment, Alice need only send Bob the original bit $b$ and the permutations $\phi_1, \ldots, \phi_m$. Bob then calculates the $b_i$ and checks that $H_i = \phi_i G_{b_i}$ for all $i$. For Alice to successfully change her commitment, she would have to know isomorphisms $G_0 \simeq H_i$ and $G_1 \simeq H_i$, which would imply that she had calculated $\psi$—an impossibility, since we had assumed the problem was hard.

# 8 ZK=IP=PSPACE

Initially, **ZK** was defined as a subclass of **IP**. Since we have **ZK** $\subseteq$ **IP**, and since the restriction on **ZK** seemed so demanding, it would come as something of a shock to find that **IP** $\subseteq$ **ZK** as well. A surprising and for some reason little-known result from Ben-Or et. al. shows that, given the same assumptions used to prove **NP** $\subseteq$ **ZK**, we can show **IP** = **ZK**.

To begin, we will have to define two new spaces of languages. The first, called **PSPACE**, consists of all languages which can be tested in polynomial *space* (as opposed to polynomial *time*.) The second, which we will call **AMC** for reasons which will become somewhat clearer, is a bit more difficult to define.

Suppose we have a set of protocols like those used in **IP**. The only difference is that Victor's random numbers are public in these protocols; that is, Peggy can see them. This would destroy some of our protocols, such as our interactive protocol for any co-RSR language, since the soundness of these protocols depends on the fact that Peggy does not know the random bits Victor generates.

[4] The proof that $G_0 \simeq G_1$ is not mentioned in [2]. However, I believe it may be necessary; otherwise, if Bob gives Alice $G_0 \not\simeq G_1$, he may have a way of differentiating graphs isomorphic to $G_0$ from those isomorphic to $G_1$, which would allow him to predict bits before Alice reveals them.

Further, since Peggy can simply compute for herself any results that Victor calculates on the random bits, it does no harm to let the message from Victor to Peggy consist solely of the random bits that Victor generates. Such protocols with public coins have historically been called Arthur-Merlin protocols, with Arthur as the verifier and Merlin as the prover. Furthermore, we require that these protocols be one-sided; that is, that valid instances of a language are always accepted. The set of languages that can be recognized by such protocols will be labeled **AMC**, for Arthur-Merlin with strict Completeness. Note that our RSR protocols fall into this category. Since any Arthur-Merlin protocol is also a valid interactive protocol, we have $\mathbf{AMC} \subseteq \mathbf{IP}$.

Our first task will be to prove that $\mathbf{IP} = \mathbf{PSPACE} = \mathbf{AMC}$. The result $\mathbf{IP} = \mathbf{PSPACE}$ is actually well-known, but we will take a slightly roundabout method of establishing this statement. First, we will show that $\mathbf{IP} \subseteq \mathbf{PSPACE}$. We may consider any interactive protocol as a binary tree with height polynomial in the length of the input. At each internal node of the tree, we have only a label dictating who should send the next bit, Peggy or Victor. At each leaf is an outcome—"accept" or "reject"—which may be calculated in polynomial time given the path from the root of the tree to the leaf in question.

It turns out that one may calculate in polynomial space the probability that the protocol will succeed. If the protocol requires $k$ rounds, the space we will need will include:

1. enough space to contain one complete path through the tree, which will be a polynomial amount;
2. enough space to contain a random bitstream for any particular response on Victor's part; and
3. $k$ counters of polynomial length, to contain probabilities.

If the probability of accepting an input is very high, the input is an instance of the language; it the probability is very low, the input is not an instance. Thus any language in **IP** can be checked in polynomial space, or $\mathbf{IP} \subseteq \mathbf{PSPACE}$. Technical details can be found in *Omer's paper—references???*. A similar proof shows that the prover in any interactive protocol may be replaced by a deterministic **PSPACE** prover.

Second, we can show that $\mathbf{PSPACE} \subseteq \mathbf{AMC}$ by abusing presently held knowledge. The fact that $\mathbf{PSPACE} \subseteq \mathbf{IP}$ is established, for example *here—does anyone have references?* by creating an interactive protocol for a **PSPACE**-complete language. If one looks closely at the protocol created, however, one sees that said protocol is in fact a one-sided Arthur-Merlin protocol. Thus one receives the proof that $\mathbf{PSPACE} \subseteq \mathbf{AMC}$ essentially for free.

So we now have $\mathbf{AMC} \subseteq \mathbf{IP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{AMC}$, which proves that the three are equal. The problem of proving $\mathbf{IP} \subseteq \mathbf{ZK}$ is conceptually easier as well, since we can now think of it as the problem of proving $\mathbf{AMC} \subseteq \mathbf{ZK}$.

Given a one-sided Arthur-Merlin protocol, we will construct a two-stage zero-knowledge protocol. In the first stage, Victor and Peggy carry out the Arthur-Merlin protocol, except that, instead of sending Merlin's answers, Peggy sends bit-commitments of Merlin's answers. In the second stage, Peggy convinces Victor that he would have accepted Peggy's answers, had he known them.

The reason that we use Arthur-Merlin protocols is now clear; if we did not, Victor would never be able to respond during the first stage of the zero-knowledge protocol. The reason for one-sidedness is a bit more subtle; essentially, if Victor cheated by sending nonrandom strings, he would conceivably gain the knowledge that Peggy accepts or rejects certain strings. The whole issue can be sidestepped by using one-sided protocols.

To be more explicit, suppose we have an Arthur-Merlin protocol of $2k$ rounds. In the first round, Arthur sends a random bitstring $r_1$; in the second, Merlin sends a response $v_1$; Arthur sends $r_2$; Merlin sends $v_2$; etc., until Arthur and Merlin send $r_k$ and $v_k$.

The first part of the zero knowledge protocol proceeds as follows, over $k$ rounds: on round $i$,

1. Victor sends a random bitstring $r_i$.
2. Peggy calculates $v_i$ and generates a random bitstring $d_i$. She calculates the commitment $f_i = f(v_i, d_i)$ and sends $f_i$ to Victor.

In the second part, Peggy has the task of convincing Victor that the $v_i$s are convincing responses without revealing any of the $v_i$s. In other words, she has to prove the following statement:

> There exist $v_1, \ldots, v_k, d_1, \ldots, d_k$ where $f_i = f(v_i, d_i)$ for each $i$, $1 \leq i \leq k$, and Arthur accepts on $r_1, v_1, \ldots, r_k, v_k$.

This statement is a member of the language SAT whose length grows polynomially with the length of the input. Since SAT $\in$ **NP**, there exists a zero-knowledge protocol for proving SAT. Thus Peggy can indeed prove to Victor that such values exist without revealing these values!

Modulo some technical details (for example, Victor's responses must all be of the same length, and the composition of two zero-knowledge protocols must be shown to be zero-knowledge) this shows that, if secure bit-commitment exists, **ZK** = **IP** = **PSPACE**.

# References

[1] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, J. Kilian, S. Micali and P. Rogaway. *Everything Provable is Probable in Zero-Knowledge.* (1988). *Advances in Cryptology—CRYPTO '88*, pp. 37–56.

[2] I. B. Damgård. *On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs.* (1989). *Advances in Cryptology—CRYPTO '89*, pp. 17–27.

[3] A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. *On Monotone Formula Closure of SZK.* (1994). *IEEE 35th Symposium on Foundations of Computer Science*, pp. 454–465.

[4] Y. Desmedt and M. Bermester. *An Efficient Zero-Knowledge Schemen for the Discrete Logarithm Based on Smooth Numbers.* (1991). *Advances in Cryptology—ASIACRYPT '91*, pp. 360–367.

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

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