Locally Random Reductions: Improvements and Applications [*]

D. Beaver[†] J. Feigenbaum[‡] J. Kilian[§] P. Rogaway[¶]

September 1, 1995

Abstract

A (t, n)-locally random reduction maps a problem instance x into a set of problem instances $y_1, \ldots, y_n$ in such a way that it is easy to construct the answer to x from the answers to $y_1, \ldots, y_n$, and yet the distribution on t-element subsets of $y_1, \ldots, y_n$ depends only on $|x|$. In this paper we formalize such reductions and give improved methods for achieving them. Then we give a cryptographic application, showing a new way to prove in perfect zero knowledge that committed bits $x_1, \ldots, x_m$ satisfy some predicate Q. Unlike previous techniques for such perfect zero-knowledge proofs, ours uses an amount of communication that is bounded by a fixed polynomial in m, regardless of the computational complexity of Q.

1 Introduction

We develop and apply a new type of reduction, which we call a *locally random reduction.* We begin with some historical motivation and context for our work. Next, we present an improved construction of locally random reductions. Finally, we apply these reductions to *zero-knowledge proofs on committed bits.*

1.1 Motivation and historical context

The notion of reducibility among computational problems has long had a pervasive influence on the theory of computation. To analyze the average case complexity of a problem, it often suffices to reduce an arbitrary instance of the problem to a random instance. For example, let p be a prime and α be a generator of Z_p^*. One can reduce the problem of computing $\log_\alpha x \bmod p$, where $x \in Z_p^*$, to that of computing $\log_\alpha y \bmod p$, where y is distributed uniformly over Z_p^*. Simply choose r uniformly at random from $\{1, \ldots, p-1\}$, compute $y = \alpha^r x \bmod p$, and let $\log_\alpha x =$

[*]These results were presented in preliminary form at the 10th Annual Crypto Conference, Santa Barbara CA, August 1990.

[†]Transarc Corporation, 707 Grant Street, Pittsburgh, PA 15219, beaver+@transarc.com. Work done at Harvard University, supported in part by NSF grant CCR-870-4513.

[‡]AT&T Bell Laboratories, Room 2C473, 600 Mountain Avenue, Murray Hill, NJ 07974-0636, jf@research.att.com.

[§]NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, joe@research.nj.nec.com. Work done at MIT and Harvard University, supported by an NSF postdoctoral fellowship.

[¶]Department of Computer Science, Engineering II Building, University of California, Davis, CA 95616, rogaway@cs.ucdavis.edu.

$(\log_\alpha y) - r \bmod (p-1)$. Thus, one can generate a "hard" instance of x by choosing x at random: If computing $\log_\alpha x$ were easy for a randomly chosen x, then it would be easy for any value of x.

More generally, suppose one could randomly reduce computing $f(x)$, where $|x| = m$, to computing $g(y)$, such that y is distributed according to some probability measure R_m. Then the average-case complexity of computing $g(y)$, where the average is computed with respect to R_m, is as high as the worst-case complexity of computing $f(x)$.

Unfortunately, this approach is limited, because of the following result. Suppose that $\Sigma_3^{\mathrm{P}} \neq \Pi_3^{\mathrm{P}}$, and that f is NP-hard. Then there is no polynomial-time random reduction from f to any function g such that the distribution on random instances y depends only on $|x|$ (cf. [1]). This result holds for a generalized notion of random reductions, known as *single-oracle instance-hiding schemes*. These schemes have a probabilistic polynomial-time bounded player P and an unbounded player O that always answers correctly. P wishes to compute $f(x)$ for some function f and an input x. P is allowed to flip coins and to interact with O for an arbitrary number of rounds but is not allowed to reveal anything more than $|x|$ to O. Here, "revealing only $|x|$ to O" means that if $|x_1| = |x_2|$, then O's views of the conversation when $x = x_1$ and when $x = x_2$ are identically distributed. A more precise and general formulation of this idea may be found in [1].

Rivest [17, 1] proposed the more general notion of *multi-oracle instance-hiding schemes*, in which P is allowed to interact with a number of oracles $O_1, \ldots, O_n$. P is not allowed to reveal more than $|x|$ to any *single* oracle O_i, but two or more oracles together may have enough information to reconstruct x completely. Whereas schemes with only one oracle appear relatively weak, Beaver and Feigenbaum proved the following theorem for multi-oracle schemes.

Theorem: [3] For any function f, there exists an $(|x|+1)$-oracle instance-hiding scheme that reveals at most $|x|$.

Because any function $f : \{0,1\}^m \longrightarrow \{0,1\}$ can be trivially reduced to a function $g : \{0,1\}^{m-c} \longrightarrow \{0,1\}^{2^c}$, the factor of $|x|+1$ may be reduced to a factor of $|x| - c \lg |x|$. In fact, we will later show how to reduce this to $|x|/c \lg |x|$.

Lipton [15] translated the arguments of [3] into the language of multivariate polynomials and applied them to the area of program testing. This framework is much easier to work with than the original framework, which involved multi-party computations on shared secrets, and furthermore allows one to prove useful program-testing results for multivariate polynomials of low degree. It has been observed that Lipton's program-testing reductions imply average-case complexity results, such as the following theorem on computing permanents over finite fields.

Theorem: [15] Let F be a finite field with more than $m+1$ elements. Suppose that, for some probabilistic polynomial-time algorithm P, and for M chosen uniformly from $m \times m$ matrices,

$$\Pr[P(M) = \mathrm{perm}(M)] > 1 - \frac{1}{3(m+1)}.$$

Then there exists a probabilistic polynomial-time algorithm Q such that, for all $m \times m$ matrices M,

$$\Pr[Q(M) = \mathrm{perm}(M)] > 1 - 2^{-m}.$$

Taking the contrapositive, if computing permanents over large finite fields is difficult in the worst case, it must also be difficult for an $\Omega(1/m)$ fraction of the instances. Since the results of

Beaver-Feigenbaum [3] and Lipton [15] appeared, a number of researchers have used random-self-reducibility properties of multivariate polynomials to show, among other things, that $P^{\#P} \subseteq$ IP (cf. [16]), IP = PSPACE (cf. [19]), and MIP = NEXPTIME (cf. [2]). A detailed overview of the relationship of locally random reductions to other basic concepts in complexity theory can be found in [9].

1.2 Our results

In this paper, we provide a formal definition of *locally random reductions*, exhibit an improved general construction of such reductions, and apply them to zero-knowledge proof systems. Informally, a (t, n)-locally random reduction from a function f to a function g works as follows. To compute $f(x)$, we use x and a string r of "random coin-flips" to generate $y_1, \ldots, y_n$. Here n, as well as t, depends only on $m = |x|$. We recover $f(x)$ by computing simple function of x, r, and $g(y_1), \ldots, g(y_n)$. Moreover, for any x_0 and x_1 of the same length m, for any $i_1, \ldots, i_t$, the distribution $y^0_{i_1}, \ldots, y^0_{i_t}$ induced by x_0 is identical to the distribution $y^1_{i_1}, \ldots, y^1_{i_t}$ induced by x_1. We prove the following theorem, which is stated informally here; a formal statement and proof are given in Section 3.

Theorem 1: For any function $f : \{0,1\}^m \to \{0,1\}$ and any constant $c > 0$, there is a function g such that f is $(t, tm/c \lg m)$-locally random reducible to g.

This improves on the results of [3, 15] mentioned above.

We apply locally random reductions in a novel protocol for *zero-knowledge proofs on committed bits*. Zero-knowledge proof systems, as originally formulated by Goldwasser, Micali, and Rackoff [13], are two-party protocols in which the parties have a common input x, and one party (the prover) convinces the other (the verifier) that, say, $f(x) = 1$, without revealing anything about x except that $f(x) = 1$. We consider a related setting in which the prover publishes a *commitment* to its private input x and then at some later time proves in zero-knowledge to the verifier that $f(x) = 1$. Furthermore, f may be unknown at the time x is committed.

We consider how to implement such proofs in the presence of an *ideal commitment scheme*. Both prover and verifier have unlimited computational power, no complexity-theoretic assumptions are made, and an ideal bit commitment scheme is assumed as a primitive. A natural question to ask is whether one can actually perform zero-knowledge proofs on committed bits in this setting. This question has been answered in the affirmative by several researchers (e.g., [5, 18]); a written account of a more recent scheme appears in [6].

It is natural to ask whether an interactive proof system is at all interesting if it requires the verifier as well as the prover to have unlimited computational power. The answer is yes, for the following reason: We are focusing on the *communication cost* of proving the value of a predicate on a set of committed bits. It is not at all clear (and might even be counterintuitive) that an arbitrary predicate f can be proven in a communication-efficient manner, even if both prover and verifier have enough computational power to compute f. All previous schemes for zero-knowledge proofs on committed bits, including those of [5, 6, 18], have bit complexity proportional to the circuit complexity of f, where by "bit complexity" we mean the total number of bits committed to or communicated between the two players. Thus, if f is an arbitrary predicate on m bits, a zero-knowledge proof that $f(x) = 1$ will require exponential communication if one uses the protocols of [5, 6, 18], *regardless of the amount of computational power one allows the verifier.* By applying locally random reductions, we achieve a protocol whose total communication cost is polynomial, even if the circuit complexity of f is exponential.

Theorem: Given an ideal commitment scheme, there exist protocols for committing and decommitting bits and a protocol for proving arbitrary predicates on a set of committed bits. The proof system reveals nothing about the committed bits other than what is implied by the predicate being true. Furthermore, the bit complexity of the proof system is polynomial in the number of input bits to the predicate – it is independent of the predicate's computational complexity.

A formal statement and proof of this theorem appears in Section 4.

Although the fact that both prover and verifier in our protocol have unlimited computational power does not detract from the theoretical importance of the fact that the protocol's communication costs are polynomial, it does render the protocol impractical. With respect to practical applicability, our protocol is not an improvement over those of [5, 6, 18].

The rest of the paper is organized as follows. In Section 2, we formally define locally random reductions and other notions that we will use later in the paper. In Section 3, we give our improved construction of locally random reductions. In Section 4, we give our communication-efficient protocol for zero-knowledge proofs on committed bits. Open questions are given in Section 5.

These results first appeared in our Technical Memorandum [4].

2 Preliminaries

2.1 Locally random reductions

We now formalize the intuition of Section 1.2.

Definition 1 Let $f : D \rightarrow \{0,1\}^*$, $g : D' \rightarrow \{0,1\}^*$, and $t, n : \mathsf{N} \rightarrow \mathsf{N}$. We say that f is **(t,n)-locally random reducible** to g in time $Q(m)$ if there is a polynomial $\rho(m)$ and a pair of $Q(m)$-time computable functions (*scatter*, *reconstruct*) such that:

- **[Correctness]** For all $m \in \mathsf{N}$ and $x \in D \cap \{0,1\}^m$, for at least 3/4 of all $r \in \{0,1\}^{\rho(m)}$,

$$f(x) = \mathit{reconstruct}(x, r, g(y_1), \ldots, g(y_{n(m)})),$$

  where $\langle y_1, \ldots, y_{n(m)} \rangle = \mathit{scatter}(x, r)$.

- **[Local randomness]** For all $m \in \mathsf{N}$ and $\{i_1, \ldots, i_{t(m)}\} \subseteq \{1, \ldots, n(m)\}$, if r is chosen uniformly at random from $\{0,1\}^{\rho(m)}$, then, for any $x_1, x_2 \in D \cap \{0,1\}^m$, the distribution on $\langle y_{i_1}, \ldots, y_{i_{t(m)}} \rangle$ induced by $\mathit{scatter}(x_1, r)$ is identical to that induced by $\mathit{scatter}(x_2, r)$.

More succinctly, we will write "f is (t,n)-lrr to g." When we omit mention of Q, it means that $Q(m)$ is a polynomial but that the specific polynomial involved is unimportant for the result under discussion. In the special case in which $f = g$, we say that f is "(t,n)-locally random self-reducible."

Informally, if T is a subset of the target instances $\{y_1, \ldots, y_{n(m)}\}$, and $|T| \leq t(m)$, then T leaks no information about the original instance x, except its length m.

2.2 Function arithmetization

A powerful technique for dealing with a Boolean function $f : \{0,1\}^m \to \{0,1\}$ is to treat f as a multivariate polynomial P over some finite field F. In this way, algebraic properties of polynomials can be directly exploited. Such *arithmetization* of Boolean functions is an important insight of Ben-Or, Goldwasser, and Wigderson [7]. The polynomial P is sometimes referred to as a "multilinear extension of f over F" (e.g., in [2, 16, 19]).

Fix a function $f : \{0,1\}^m \to \{0,1\}$ and a finite field F. We use α_1^i to denote the polynomial x_i and α_0^i to denote the polynomial $1 - x_i$. (The "1" is the multiplicative identity of F.) Given an m-bit string $a = a_1 \dots a_m$, we define the polynomial δ_a by

$$\delta_a = \prod_{i=1}^{m} \alpha_{a_i}^i .$$

Now let the polynomial $P(x)$ be given by

$$P(x) = \sum_{a \in \{0,1\}^m} f(a)\delta_a(x) \ .$$

This is the arithmetization of f over F.

Here is an example. Let $f(x_1x_2x_3) = x_1 \oplus x_2 \oplus x_3$, where $\oplus$ denotes exclusive-or. Then the arithmetization of f is the polynomial

$$P(x_1, x_2, x_3) = x_1(1-x_2)(1-x_3) + (1-x_1)x_2(1-x_3) + (1-x_1)(1-x_2)x_3 + x_1x_2x_3.$$

At this point, we make two observations. First, in the definition of δ_a, each variable can appear at most once in the product, and so δ_a is linear in each variable x_i. Thus P is also linear in each variable x_i (being the sum of monomials that are linear in x_i). Second, for any $x \in \{0,1\}^m$, $P(x) = f(x)$. This identity may be verified by noting that, in the sum given by the definition of $P(x)$, all the terms are 0 except for one that is equal to $f(x)$.

Throughout this paper, we assume that the finite field F has characteristic two. This allows us to choose an element of F uniformly at random simply by flipping coins. All of our definitions and results can be stated for F of characteristic greater than two as well. Certain protocols that work with probability one when F has characteristic two may fail with exponentially small probability when F has higher characteristic, because a sequence of coin flips may fail to yield an element of F. Otherwise, everything that we present is the same for all finite fields.

3 Improved locally random reductions

We now show how to improve the results of Beaver-Feigenbaum [3] and Lipton [15]. We first exhibit a parameterized family of random-self-reductions for multivariate polynomials over sufficiently large finite fields. We then give, for any constant $c > 0$ and any m-bit function f, a $(t, t\lfloor m/c \lg m \rfloor)$-locally random reductions from f to some other function g.

Lemma 1 There is a polynomial $Q(m)$ having the following property. Let d and t be numbers, and let F be a finite field of at least $dt + 2$ points. Let $P(x_1, \dots, x_m)$ be a polynomial in $F[x_1, \dots, x_m]$

of total degree at most d. Then P is locally random self-reducible in time $Q(m+d+t+\lg|F|)$. Furthermore, there is a single pair of functions (*scatter*, *reconstruct*) that serves as a locally random self-reduction for any P satisfying the above conditions.

Proof: Our proof proceeds along the lines of [3], using the polynomial framework of [15]. First, we define $\mathit{scatter}(X, r)$. Let $X = (x_1, \ldots, x_m) \in F^m$, and regard r as a set of mt random elements of F, denoted $\{c_{i,j}\}$, where $1 \leq i \leq m$ and $1 \leq j \leq t$. Let $\alpha_1, \ldots, \alpha_{dt+1}$ denote distinct nonzero elements of F. Define $p_i(z)$ by

$$p_i(z) = c_{i,t}z^t + \cdots + c_{i,1}z + x_i.$$

Finally, define

$$\mathit{scatter}(X, \{c_{i,j}\}) = (Y_1, \ldots, Y_{dt+1}),$$

where $Y_k = (p_1(\alpha_k), \ldots, p_m(\alpha_k))$.

Before describing *reconstruct*, we explain our definition of *scatter*. Define $\hat{P}(z)$ by

$$\hat{P}(z) = P(p_1(z), \ldots, p_m(z)).$$

Because P is of total degree at most d, and each $p_i(z)$ is of degree t in z, the curve $\hat{P}(z)$ is of degree at most dt. By definition,

$$\begin{aligned} P(Y_k) &= \hat{P}(\alpha_k), \text{ and} \\ P(X) &= \hat{P}(0) \qquad (\text{because } p_i(0) = x_i). \end{aligned}$$

We now define *reconstruct*. Recall that computing $P(X)$ is equivalent to computing $\hat{P}(0)$. Because $\hat{P}$ is a univariate polynomial of degree at most dt, $\hat{P}(0)$ may be recovered from $\hat{P}(\alpha_1), \ldots, \hat{P}(\alpha_{dt+1})$ by Lagrangian interpolation. More explicitly, we define *scatter* by

$$\mathit{scatter}(P(Y_1), \ldots, P(Y_{dt+1})) = \sum_{k=1}^{dt+1} t_k Y_k \ ,$$

where $t_1, \ldots, t_{dt+1} \in F$ are constants defined by

$$t_k = \prod_{j \neq k} \frac{-\alpha_j}{\alpha_k - \alpha_j} \ .$$

Thus, (*scatter*, *reconstruct*) has the correctness property required by Definition 1, and both *scatter* and *reconstruct* can be computed in the stated polynomial number of steps. Thus it suffices to show that, for any $X_1, X_2 \in F^m$ and any sequence $(i_1, \ldots, i_t)$, the distribution on $(Y_{i_1}, \ldots, Y_{i_t})$ induced by $\mathit{scatter}(X_1, r)$ is the same as that induced by $\mathit{scatter}(X_2, r)$, i.e., that (*scatter*, *reconstruct*) has the local randomness property also required by Definition 1. We show this by using the following well known fact about polynomial interpolation: Given points $(x_1, y_1), \ldots, (x_t, y_t)$, where all the x_i's are distinct and nonzero, and fixing c_0, there is exactly one polynomial of the form $c_t z^t + \cdots + c_1 z + c_0$ that agrees with all of these points. Thus, the fact that the $c_{i,j}$'s are chosen independently and uniformly at random, combined with our definition of *scatter*, implies that, for any distinct $\alpha_{k_1}, \ldots, \alpha_{k_t} \in F - \{0\}$, and any $X, Y_{k_1}, \ldots, Y_{k_t} \in F^m$, there is exactly one consistent value of C. Therefore, the distribution on $(Y_{k_1}, \ldots, Y_{k_t})$ is uniform over $(F^m)^t$, for any value of $X \in F^m$. ▌

Beaver and Feigenbaum showed that for, any m-bit boolean function f, there is a function g such that f is $(1, m+1)$-locally random reducible to g. We now show how to reduce the total number of queries from $m+1$ to $\lfloor m/c \lg m \rfloor$, for any constant $c > 0$.

Theorem 1 Fix a constant $c > 0$ and a function $t : \mathsf{N} \to \mathsf{N}$. Then there is a polynomial $Q(m)$ having the following property: For any function $f : \{0,1\}^m \to \{0,1\}$ there is a function g such that f is $(t, t\lfloor m/c \lg m)\rfloor)$-locally random reducible to g in time $Q(m)$.

Proof: Let F be a finite field of the form $\mathrm{GF}(2^l)$, where $l \geq \lceil mt \rceil$. We first show how to reduce the computation of the arithmetization $P(x_1, \ldots, x_m)$ of f over F to the computation of another multivariate polynomial $P^*(y_1, \ldots, y_v)$ over F of total degree at most $\lfloor m/c \lg m \rfloor$. We then apply Lemma 1 to complete our proof. Partition the set $\{1, \ldots, m\}$ into disjoint subsets $S_1, \ldots, S_d$, each of size at most $c \lg m + 1$. For any i and any nonempty $T \subseteq S_i$, we define a new variable y_T, given by

$$y_T = \prod_{i \in T} x_i.$$

Let $I = \{i_1, \ldots, i_k\}$ be any subset of the indices $\{1, \ldots, m\}$ and $ax_{i_1} \cdots x_{i_k}$ a monomial in which each variable appears at most once. We can transform this degree $k \leq m$ monomial into a monomial of degree $\leq d$ via the mapping

$$ax_{i_1} \cdots x_{i_k} \longrightarrow a \prod_{i=1}^{d} y_{I \cap S_i}.$$

It is easy to verify that the values of the two monomials are equal, given the above change of variables. Because the arithmetization P of f is a sum of monomials in which each variable appears once, transforming each monomial of P as above yields a new polynomial P^* of degree at most d. Finally, one can rename the subscripts taken by our variables y_T to be integers instead of sets. This purely syntactic transformation will sometimes be made for notational reasons, allowing us to say $y_1, \ldots, y_v$ when convenient, but it is otherwise unnecessary. We can easily bound v, the number of variables in P^*, by

$$v \leq \left\lfloor \frac{m}{c \lg m} \right\rfloor \cdot 2^{c \lg m + 1} \leq \frac{2m^{c+1}}{c \lg m}.$$

Here is a simple example of the change of variables, with $m = 6$ and $d = 3$. Suppose that

$$P(x_1, \ldots, x_6) = x_1 x_2 x_3 x_4 x_5 x_6 - x_2 x_3 x_6 + 2 x_1 x_2 x_3 x_6.$$

First, let $S_1 = \{1,2\}$, $S_2 = \{3,4\}$, and $S_3 = \{5,6\}$, yielding variables

$$y_{\{1\}}, y_{\{2\}}, y_{\{1,2\}}, y_{\{3\}}, y_{\{4\}}, y_{\{3,4\}}, y_{\{5\}}, y_{\{6\}}, y_{\{5,6\}}.$$

The polynomial P^* is given by

$$P^*(y_{\{1\}}, \ldots, y_{\{5,6\}}) = y_{\{1,2\}} y_{\{3,4\}} y_{\{5,6\}} - y_{\{2\}} y_{\{3\}} y_{\{6\}} + 2 y_{\{1,2\}} y_{\{3\}} y_{\{6\}}.$$

Note that it may be infeasible to write down P or P^*, because the number of terms in one of both may be exponential in m. However the reduction from P to P^* only requires computing the new variables $\{y_T\}$, which can be done with a small number of multiplications in our field. For example, $y_{\{3,4\}}$ is computed by multiplying x_3 and x_4.

We now define our reduction (*scatter*, *reconstruct*). On input $X = x_1, \ldots, x_m$, $\mathit{scatter}(X, r)$ first computes $x_1, \ldots, x_m \in F$, where boolean 0's are transformed into the 0 element in F, and boolean 1's are transformed into the 1 element in F. This trivial transformation effects the reduction from f to P. Next, *scatter* computes the variables $\{y_T\}$, effecting the reduction from P to P^*. Note that $P^*(y_1, \ldots, y_v) = f(x_1, \ldots, x_m)$, where 0 and 1 field elements are identified with 0 and 1 boolean values. Finally, *scatter* performs the mapping used by the $(t, dt + 1)$-locally random self-reduction given in Lemma 1, for v-variable polynomials over F of degree $d = \lfloor m/c \lg m \rfloor$.

We define *reconstruct* to be the same as in Lemma 1, except that it interprets 0 and 1 field elements as their boolean equivalents.

By Lemma 1, our reduction (*scatter*, *reconstruct*) always give the correct answer. Furthermore, the number of algebraic operations performed by *reconstruct* and *scatter* is bounded by some polynomial in v and t. Because v is bounded by some polynomial in m (depending on c), and the requisite field operations can be implemented in time polynomial in m and t, the total number of bit operations performed by *reconstruct* and *scatter* is polynomial in m and t. ∎

4 Zero-knowledge proofs on committed bits

In this section, we formally define ideal bit commitment schemes and review the notion of zero-knowledge proofs on committed bits. In the protocols we will describe, there is one party (the "prover") who commits to a set of bits and later proves assertions about these committed bits, and there is another party (the "verifier") who verifies the proofs on the committed bits.

Intuitively, we think of an ideal commitment scheme as having physical envelopes that the prover can fill with information and place on the table. If the prover later opens an envelope, the verifier knows its contents have not been changed.

We are interested in the notion of *zero-knowledge proofs on committed bits*. Such commitments have also been referred to as *notarized envelopes*. That is, one would like to commit to a set of bits $b_1, \ldots, b_m$ and at some later time prove some predicate $Q(b_1, \ldots, b_m)$ on these bits, without revealing the values of $b_1, \ldots, b_m$ or other information not implied by $Q(b_1, \ldots, b_m)$.

Ideal commitment schemes were used in the construction of zero-knowledge proofs for predicates in NP (cf. [11]) and IP (cf. [14]). Zero-knowledge proofs on committed bits were first used in the study of multi-party secure computation [12] and were based on complexity theoretic assumptions. Simple schemes for basing zero-knowledge proofs on committed bits on ideal commitment schemes were developed not long thereafter (e.g., [5, 18]) but did not appear in the literature until [6]. These schemes allowed one to prove arbitrary predicates on k committed bits using a total amount of communication that was potentially exponential in k.

This exponential communication cost is sometimes acceptable. For example, one can transform any NP predicate $Q(b_1, \ldots, b_m)$ into a predicate of the form

$$(\exists y_1, \ldots, y_l) Q'(b_1, \ldots, b_m, y_1, \ldots, y_l),$$

where, $Q' = \wedge_i Q'_i$ and each Q'_i is a predicate on just three variables, then prove $Q(b_1, \ldots, b_m)$ in zero knowledge by committing to suitable values for $y_1, \ldots, y_l$ and proving each of the predicates Q'_i in zero-knowledge. However, such a transformation cannot be applied to arbitrary predicates and can be very unwieldy even for NP predicates. This technique also leaves open the question of whether the communication cost of zero-knowledge proofs on committed bits depends intrinsically on the computational complexity of the predicate to be proven. In the remainder of this section, we answer this fundamental question in the negative.

4.1 Formal definitions

In this section we describe the model of computation for interactive proofs in the presence of an ideal commitment scheme. We then go on to define (perfect) zero-knowledge proofs in this model.

4.1.1 Ideal commitment schemes

An ideal commitment scheme (ICS) can be thought of as a special type of channel that connects the prover P to the verifier V. When we run the protocol specified by P and V, any string that V writes down for P will be delivered (unmodified) to P; but messages sent from P to V are transmitted in the following way. Initialize $S \leftarrow \emptyset$ and then:

1. When P transmits on its channel to V a message
$$\mathrm{commit}(x, t),$$
if there is no ordered pair $(x', t) \in S$, then we set $S \leftarrow S \cup \{(x, t)\}$ and deliver to V the message t. If there is already an $(x', t) \in S$ then the empty string is delivered to V.

2. When P transmits on its channel to V a message
$$\mathrm{decommit}(t),$$
if there is some pair $(x, t) \in S$ then we deliver to V the message (x, t). If there is no such pair $(x, t) \in S$, then the empty string is delivered to V.

We could have provided P a "direct" channel to V, but this is trivially simulated with the channel above.

We say that P *commits to* $x = x_1 \ldots x_m$ if P transmits in the course of the protocol:
$$\mathrm{commit}(x_1, \texttt{x-bit-1}), \ldots, \mathrm{commit}(x_m, \texttt{x-bit-}m), (0, \texttt{length-x=}m)\ .$$

We say that P *reveals* x if it sends the corresponding decommitments.

We will use the notation $\texttt{Channel}^S_{P \to V}(x)$ to denote the message delivered when x is transmitted on the $P \to V$ channel, which is currently in state S. Note that this operation has a side effect on S.

4.2 ICS protocol execution

We model players P and V as $\{0,1\}^*$-valued functions on initial input $s \in \{0,1\}^*$, (verifier) view $z \in \{0,1,\#\}^*$, and coins flips $r \in \{0,1\}^\infty$. An $R(m)$-round execution $(P(s_1), V(s_2))$ is defined by the following experiment: set $z \leftarrow \lambda$; set $S \leftarrow \emptyset$; choose random strings $r_1, r_2 \in \{0,1\}^\infty$; then:

```
for $i \leftarrow 1$ to $R(|x|)$ do
    $z \leftarrow z \,\#\, \mathtt{Channel}^S_{P \to V}(P(s_1, z, r_1))$
    $z \leftarrow z \,\#\, V(s_2, z, r_2)$
```

We say that an execution of $(P(s_1), V(s_2))$ *accepts* (or simply V *accepts*) if the last bit of the final value of z is 1; else we say it *rejects*. The (verifier's) *view* is the random variable that gives s_2, r_2, and the final value of z. The *communication complexity* of an execution is the length of the final value of z.

4.2.1 Zero-knowledge proofs on committed bits

A zero-knowledge proof that predicate Q holds on committed bits $x_1, \ldots, x_m$ is a like a neutral third party that does nothing but check that $Q(x_1, \ldots, x_m) = 1$, reporting the answer back to V. Nothing else is revealed.

As in the more customary setting of Goldwasser, Micali and Rackoff [13], we can formalize this idea by using a *simulator*: We require of any (possibly cheating) verifier that there be an algorithm that produces a distribution on (fake) views that coincides with the distribution on (real) views received by that verifier (when interacting with the prover who has initial input x, where $Q(x) = 1$). By effectively demonstrating that the verifier could have computed its view on its own (knowing nothing but $Q(x) = 1$), the existence of the simulator assures us that the verifier learns no more than it should.

Another way to model potential information leakage follows the notion of "witness indistinguishability" of Feige and Shamir [8]. In particular, for any equal length x and x' that satisfy predicate Q, the views that the verifier gets in these cases should be identical. This approach concerns itself more with hiding the input than with leaking extraneous information.

In the formalization we now give, we follow the second approach. Equivalent definitions can be formulated using simulators.

Definition 2 An $R(m)$-round, $\epsilon(m)$-error **ICS proof system** for predicate Q is a pair of players (P, V) such that:

- (Completeness) For any x such that $Q(x) = 1$, an $R(|x|)$-round execution $(P(x), V(|x|))$ accepts, and in it P commits to x.
- (Soundness) For any player $\tilde{P}$ that commits to its initial input x, if $Q(x) = 0$, then the $R(|x|)$-round execution $(\tilde{P}(x), V(|x|))$ accepts with probability at most $\epsilon(|x|)$.

When $R(m)$ is polynomial and $\epsilon(m) < 1/2$ is constant we omit mention of these parameters. If the communication complexity is bounded by a polynomial in m, we say that (P, V) is *communication-efficient.*

Definition 3 An ICS proof system (P, V) for predicate Q is **zero-knowledge** if, for all $\tilde{V}$ and all x_1, x_2 such that $|x_1| = |x_2|$ and $Q(x_1) = Q(x_2) = 1$, the view of $(P(x_1), \tilde{V}(|x_1|))$ is identical to the view of $(P(x_2), \tilde{V}(|x_2|))$.

4.3 A communication-efficient protocol for proofs on committed bits

We now present a new protocol, based on an ICS, for performing zero-knowledge proofs on committed bits. In our protocol, a computationally unbounded prover P can prove arbitrary predicates in zero-knowledge to a computationally unbounded verifier V. Unlike the previous protocols, our protocol requires communication that is only polynomial in the number of committed bits, regardless of the circuit complexity of the predicate being proven. Note that the verifier must be computationally unbounded, because it must verify arbitrary predicates.

For our discussion, we will often blur the distinction between boolean values and the 0 and 1 elements of a finite field. First, we use a standard trick of representing each bit to be committed as a random exclusive-or of two bits (equivalently, a random sum over $GF(2)$). The following simple protocols are used to commit and reveal bits.

Protocol COMMIT$(x_1, \ldots, x_m)$ For $1 \leq i \leq m$, P uniformly chooses $x_i^0, x_i^1 \in \{0, 1\}$, subject to $x_i = x_i^0 \oplus x_i^1$, and commits to x_i^0 and x_i^1 using the ICS.

Protocol REVEAL(i) The prover reveals x_i^0 and x_i^1 using the ICS. V computes $x_i = x_i^0 \oplus x_i^1$.

It is easy to verifier that the value of a bit recovered during the REVEAL protocol must be the same as that during the COMMIT protocol. Furthermore, as soon as P has committed to x_i^0 and x_i^1, he has implicitly committed to a bit x_i that is guaranteed to be well defined. The issue of committed bits' being well defined will arise later but can be safely ignored at this point.

Our protocol for performing zero-knowledge proofs on a set of committed bits is based on the reduction given in the proof of Lemma 1, where $t = 1$. In order to prove the boolean predicate $Q(x_1, \ldots, x_m)$, P and V first arithmetize Q, as in the proof of Theorem 1 (treating $Q(X)$ as a boolean function that is 1 iff $Q(X)$ is true). For the rest of the protocol, P must show that $Q^*(x_1, \ldots, x_m) = 1$, where Q^* is a degree $\leq m$ multivariate polynomial over a finite field F. F must have at least $m + 1$ distinct nonzero elements, denoted $\alpha_1, \ldots, \alpha_{m+1}$.

The zero-knowledge proof proceeds in two phases. In the *commitment* phase, P generates a run of the $(1, m + 1)$-locally random self-reduction on Q^*, "breaks" the computation into random pieces, and commits to these pieces. In the *challenge* phase, V randomly chooses to see certain pieces of the reduction and uses this glimpse to verify probabilistically that the self-reduction was honest.

In the commitment phase, P uniformly generates $\{c_i \in F\}$ and then follows the reduction in Lemma 1 to generate $\{y_{i,j}\}$. He then computes

$$\begin{aligned} Y_j &= (y_{1,j}, \ldots, y_{m,j}) \\ z_j &= Q^*(Y_j), \end{aligned}$$

and finally reconstructs the final answer,

$$w = \sum_{j=1}^{m+1} t_j z_j,$$

Protocol PROVE$(x_1, \ldots, x_m, Q)$ [Commitment Stage]
Let F be a finite field with at least $m+2$ element, and let Q^* be the arithmetization of Q over F. Let $\alpha_1, \ldots, \alpha_{m+1} \in F$ be distinct and nonzero. Define $t_1, \ldots, t_{m+1} \in F$ by $t_j = \prod_{i \neq j} \frac{-\alpha_i}{\alpha_j - \alpha_i}$.

1: For $1 \leq i \leq m$ and $1 \leq j \leq m+1$, P uniformly chooses $c_i \in F$, and computes:

$$\begin{aligned} y_{i,j} &= x_i + c_i \alpha_j, \\ Y_j &= (y_{1,j} \ldots, y_{m,j}), \\ z_j &= Q^*(Y_j), \text{ and,} \\ w &= \sum_{j=1}^{m+1} t_j z_j. \end{aligned}$$

2: P uniformly chooses $c_i^0, c_i^1 \in F$, subject to $c_i = c_i^0 + c_i^1$, and $z_j^0, z_j^1 \in F$, subject to $z_j = z_j^0 + z_j^1$. P then computes:

$$\begin{aligned} y_{i,j}^b &= x_i^b + c_i^b \alpha_j, \text{ and,} \\ w^b &= \sum_{j=1}^{m+1} t_j z_j^b, \end{aligned}$$

where $b \in \{0, 1\}$. Note that $y_{i,j} = y_{i,j}^0 + y_{i,j}^1$ and $w = w^0 + w^1$.

3: For $b \in \{0,1\}, 1 \leq i \leq m$, and $1 \leq j \leq m+1$, P commits to $c_i^b, y_{i,j}^b, z_j^b$ and w^b using the ICS.

Figure 1: Commitment stage of the zero-knowledge proof system.

where $t_j = \prod_{i \neq j} \frac{-\alpha_i}{\alpha_j - \alpha_i}$. After generating this run of the reduction, P breaks up each $c_i, y_{i,j}, z_j$ and w into two halves whose sum (over F) is equal to the original and then commits to each half. Thus, we have $c_i = c_i^0 + c_i^1$, $w = w^0 + w^1$, etc. We give the commitment stage of the protocol in Figure 1.

In the challenge phase of the protocol, V makes one of three general requests. He can ask P to reveal the "0 half" or the "1 half" of the self-reduction and verify a number of linear constraints. He can ask P to reveal Y_j and z_j for some j (by revealing both halves of all their relevant components) and verify that $z_j = Q^*(Y_j)$. Or he can ask P to reveal w (by revealing w^0 and w^1) and verify that $w = 1$. We give the challenge stage of the protocol in Figure 2.

4.4 Properties of our proof system

In this section, we argue that our protocols have the properties of a zero-knowledge proof system. We first show that our protocol is *complete*: If both parties behave properly, then V always accepts

Protocol PROVE$(x_1, \ldots, x_m, Q)$ [Challenge Stage]
V makes one of the following $m+4$ challenges, each with equal probability:

1. [For b either 0 or 1] For $1 \le i \le m$ and $1 \le j \le m+1$, V asks P to reveal $x_i^b, c_i^b, y_{i,j}^b, z_j^b$ and w^b. V accepts iff $y_{i,j}^b = x_i^b + c_i^b \alpha_j$ and $w^b = \sum_{j=1}^{m+1} t_j z_j^b$.
2. [For $1 \le j \le m+1$] For $b \in \{0,1\}$ and $1 \le i \le m$, V asks P to reveal $y_{i,j}^b$ and z_j^b. V then computes $\hat{Y}_j = (y_{1,j}^0 + y_{1,j}^1 \ldots, y_{m,j}^0 + y_{m,j}^1)$ and $\hat{z}_j = z_j^0 + z_j^1$ and accepts iff $\hat{z}_j = Q^*(\hat{Y}_j)$.
3. V asks P to reveal w^0 and w^1 and accepts iff $w^0 + w^1 = 1$.

Figure 2: Challenge stage of the zero-knowledge proof system.

a correct assertion. Next, we show that our protocol is weakly sound: V rejects a false assertion with probability at least $1/poly(m)$. Finally, we show that our protocol is zero-knowledge: A proof that $Q(x_1, \ldots, x_m) = 1$ conveys no extra information about $x_1, \ldots, x_m$.

Lemma 2 If $Q(x_1, \ldots, x_m) = 1$, and P and V follow PROVE$(x_1, \ldots, x_m, Q)$, then V always accepts.

Proof: It suffices to show that V accepts for each of the 3 types of challenges he might make. The first challenge is trivially satisfied, by the definition of $y_{i,j}^b$ and w^b. To show that the second challenge is satisfied, it suffices to show that the values for $\hat{z}_j$ and $\hat{Y}_j$ reconstructed by V are truly equal to those given by P. By construction, $z_j = z_j^0 + z_j^1$. The case for Y_j follows from the identity $y_{i,j} = y_{i,j}^0 + y_{i,j}^1$, which may be verified by:

$$\begin{aligned} y_{i,j}^0 + y_{i,j}^1 &= \left(x_i^0 + c_i^0 \alpha_j\right) + \left(x_i^1 + c_i^1 \alpha_j\right) \\ &= \left(x_i^0 + x_i^1\right) + \left(c_i^0 + c_i^1\right) \alpha_j \\ &= x_i + c_i \alpha_j \\ &= y_{i,j}. \end{aligned}$$

To show that the third challenge is satisfied, it suffices to show that $w = 1$ and $w = w^0 + w^1$. That $w = 1$ follows from the fact that $Q^*(x_1, \ldots, x_m) = 1$ and the fact the construction of Lemma 1 always gives the correct answer. To see that $w = w^0 + w^1$, note that

$$\begin{aligned} w^0 + w^1 &= \left(\sum_{j=1}^{m+1} t_j z_j^0\right) + \left(\sum_{j=1}^{m+1} t_j z_j^1\right) \\ &= \sum_{j=1}^{m+1} t_j \left(z_j^0 + z_j^1\right) \\ &= \sum_{j=1}^{m+1} t_j z_j \\ &= w. \quad \blacksquare \end{aligned}$$

Lemma 3 Suppose that $Q(x_1, \ldots, x_m) = 0$. Then for any (possibly malicious) $\hat{P}$, if V obeys the protocol, he rejects with probability at least $1/(m+4)$, regardless of $\hat{P}$'s strategy.

Proof: Given committed values for $c_i^b, y_{i,j}^b, z_j^b$, and w^b, define $c_i = c_i^0 + c_i^1$, $y_{i,j} = y_{i,j}^0 + y_{i,j}^1$, $z_j = z_j^0 + z_j^1$, and $w = w^0 + w^1$. Suppose that, for all i and j,

$$\begin{aligned} y_{i,j} &= x_i + c_i \alpha_j, \\ z_j &= Q^*(y_{1,j}, \ldots, y_{m,j}), \text{ and} \\ w &= \sum_{j=1}^{m+1} t_j z_j. \end{aligned}$$

Then, by the proof of Lemma 1, $w = Q^*(x_1, \ldots, x_m) \neq 1$, and V rejects if he asks to see w^0 and w^1 (a Type 3 of challenge). If $y_{i,j} \neq x_i + c_i \alpha_j$ for some i, j, then for some $b \in \{0,1\}$ it must hold that $y_{i,j}^b \neq x_i^b + c_i^b \alpha_j$, and V rejects if he makes a Type 1 challenge, for the appropriate value of b. Similarly, if $w \neq \sum_{j=1}^{m+1} t_j z_j$, for some j, then for some $b \in \{0,1\}$, $w^b \neq \sum_{j=1}^{m+1} t_j z_j^b$, and V again rejects if he makes the appropriate Type 1 challenge. Finally, if for some j, $z_j \neq Q^*(y_{1,j}, \ldots, y_{m,j})$, then V rejects if he makes a Type 2 challenge, with that value of j. Thus, in all cases, there must be at least one challenge that causes V to reject, and that challenge is chosen with probability at least $1/(m+4)$. ▮

Lemmas 2 and 3 show that the protocol is a 1-round, $(1 - \frac{1}{m+4})$-error proof system. We next show that the protocol is zero-knowledge and then discuss how to reduce the probability that a false statement is accepted.

Lemma 4 Suppose that $Q(x_1, \ldots, x_m) = Q(\bar{x}_1, \ldots, \bar{x}_m) = 1$ and that, for some set $T \subseteq \{1, \ldots, m\}$, $x_t = \bar{x}_t$ for $t \in T$. Let $\hat{V}$ be an arbitrary computationally unbounded party. Then the distribution on $\hat{V}$'s view induced by running $\textsc{commit}(x_1, \ldots, x_m)$, $\textsc{prove}(x_1, \ldots, x_m, Q)$, and $\textsc{reveal}(t)$ for $t \in T$ is identical to that induced by running $\textsc{commit}(\bar{x}_1, \ldots, \bar{x}_m)$, $\textsc{prove}(\bar{x}_1, \ldots, \bar{x}_m, Q)$, and $\textsc{reveal}(t)$ for $t \in T$.

Proof: The bulk of the proof consists of analyzing the information revealed to $\hat{V}$ during the execution of the PROVE protocol. Suppose P commits to $x_1, \ldots, x_m$ by committing to $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)$. We first show that, for each possible challenge $\hat{V}$ can make, there exists $b \in \{0,1\}$ such that his view can be generated from $x_1^b, \ldots, x_m^b$ and in no way depends on $x_1^{1-b}, \ldots, x_m^{1-b}$. Indeed, for Type 1 and Type 3 queries, one can generate $\hat{V}$'s view without looking at $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)$ at all.

If $\hat{V}$ makes a Type 1 challenge, for either value of b, his view consists of $x_i^b, c_i^b, y_{i,j}^b, z_j^b$ and w^b, for $1 \leq i \leq m$ and $1 \leq j \leq m+1$. The values of c_i^b and z_j^b are uniform over F. Furthermore, $y_{i,j}^b$ and w^b are functions of only $x_1^b, \ldots, x_m^b$, $c_1^b, \ldots, c_m^b$, and $z_1^b, \ldots, z_{m+1}^b$ (ignoring the α_i's, which are publicly known). Thus, his view from a Type 1 challenge (with value b) can be generated from only $x_1^b, \ldots, x_m^b$.

If $\hat{V}$ makes a Type 2 challenge, with a given value of j, his view consists of $y_{i,j}^b$ and z_j^b, for $b \in \{0,1\}$, and $1 \leq i \leq m$. First note that the distribution induced on (z_j^0, z_j^1) depends only on the distribution of $Y_j = (y_{1,j}, \ldots, y_{m,j})$. By the properties of our locally random reduction, $y_{1,j}, \ldots, y_{m,j}$ are independently and uniformly distributed over F, regardless of the values of $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)$. We have the identities

$$\begin{aligned} y_{i,j}^b &= x_i^b + c_i^b \alpha_j, \\ y_{i,j} &= x_i + c_i \alpha_j, \text{ and} \\ x_i &= x_i^0 + x_i^1. \end{aligned}$$

Furthermore, the c_i^b's are distributed uniformly and independently, subject to $c_i = c_i^0 + c_i^1$, and $\alpha_j \neq 0$. By a simple probability argument, the distribution on

$$y_{1,j}^0, y_{1,j}^1, \ldots, y_{m,j}^0, y_{m,j}^1$$

is uniform, subject to $y_{i,j} = y_{1,j}^0 + y_{1,j}^1$. Hence, if $\hat{V}$ makes a Type 2 challenge, his view from this challenge does not depend on the values of $x_1^0, x_1^1, \ldots, x_m^0, x_m^1$.

Finally, if $\hat{V}$ makes a Type 3 challenge, then his view consists of w^0 and w^1. We claim that w^0 and w^1 are uniformly distributed subject to $w^0 + w^1 = 1$, and thus $\hat{V}$'s view does not depend on $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)$. First, note that $w = 1 = w^0 + w^1$. Because $w^0 = \sum_{j=1}^{m+1} t_j z_j^0$, $z_1^0, \ldots, z_{m+1}^0$ are uniformly and independently distributed, and at least one value of $\{t_j\}$ is nonzero (in fact, every t_j is nonzero), it follows that w^0 is uniformly distributed.

Now, recall that the COMMIT protocol reveals nothing about the values of $x_1, \ldots, x_m$ and that the REVEAL(t) protocol releases the values of x_t^0 and x_t^1. Hence, one can always generate $\hat{V}$'s view by looking at $\{(x_t^0, x_t^1) | t \in T\}$ and $x_1^b, \ldots, x_m^b$, for some value of b that depends only on the type of $\hat{V}$'s challenge. However, if $x_1, \ldots, x_m$ and $\bar{x}_1, \ldots, \bar{x}_m$ are as in the statement of the lemma, then, for either value of b, the induced distribution on

$$\{(x_t^0, x_t^1) | t \in T\}, x_1^b, \ldots, x_m^b, \text{ and } \{(\bar{x}_t^0, \bar{x}_t^1) | t \in T\}, \bar{x}_1^b, \ldots, \bar{x}_m^b$$

is identical, and hence $\hat{V}$'s view is also identical. ∎

We can view the above argument as an algorithm for simulating $\hat{V}$'s view, knowing only x_t for $t \in T$. During the proof process, the simulator simply talks to $\hat{V}$, generating its responses according to the algorithm given in the proof. At some point, it may need to know $x_1^b, \ldots, x_m^b$ for some $b \in \{0, 1\}$, at which point the simulator uniformly generates $x_1^b, \ldots, x_m^b$ and continues. When it comes time to simulate the revelation of x_t for $t \in T$, the simulator learns these values, computes $x_t^{1-b} = x_t - x_t^b$ for $t \in T$ (choosing $x_1^b, \ldots, x_m^b$ uniformly if they have not been chosen before), and outputs the appropriate values.

One drawback to the scheme given above is the low probability that a verifier will catch an incorrect proof. This problem has been dealt with in previous protocols for zero-knowledge proofs on committed bits; the ideas used there carry over to our protocol without any conceptual alteration, and thus we simply state without proof the stronger results that we obtain using thses standard techniques.

The basic idea is to run several independent copies of the protocols. Instead of breaking each x_i into a single pair, (x_i^0, x_i^1), P will break each x_i into a sequence of independent pairs,

$$(x_i^0[1], x_i^1[1]), \ldots, (x_i^0[l], x_i^1[l]).$$

Similarly, P reveals x_i by revealing all l pairs that he previously committed. When P is honest, $x_i^0[j] \oplus x_i^1[j]$ will have the same value, x_i, for all values of j. With a malicious prover $\hat{P}$, there is no such guarantee. In this case, we define x_i to be the majority of $x_i^0[j] \oplus x_i^1[j]$, for $1 \leq j \leq l$. Under this interpretation, even a malicious prover is guaranteed to be committing to some unambiguous value.

More precisely, recall the following standard protocol that is used in earlier work on zero-knowledge proofs, e.g., in those of Bennett [5] and Rudich [18]. Given two pairs, $(x_i^0[j], x_i^1[j])$ and $(x_i^0[k], x_i^1[k])$, we wish to give a zero-knowledge proof that

$$x_i^0[j] \oplus x_i^1[j] = x_i^0[k] \oplus x_i^1[k].$$

This is accomplished by using protocol PROVE-EQUAL on the four committed bits.
Protocol PROVE-EQUAL$(x_1^0, x_1^1, x_2^0, x_2^1)$ /* Prove that $x_1^0 \oplus x_1^1 = x_2^0 \oplus x_2^1$ */

1: P sends V the value of $x_1^0 \oplus x_2^0$.

2: V uniformly chooses $b \in \{0,1\}$ and sends b to P.

3: P reveals x_1^b and x_2^b to V, who accepts iff $x_1^b \oplus x_2^b$ is equal to the value sent in Step 1.

The PROVE-EQUAL protocol is known to have the following properties:
Property 1: If $x_1^0 \oplus x_1^1 \neq x_2^0 \oplus x_2^1$, then V rejects with probability at least $\frac{1}{2}$, regardless of $\hat{P}$'s strategy.
Property 2: Let $x \in \{0,1\}$ and $x_1^0, x_1^1, x_2^0, x_2^1$ be chosen uniformly subject to

$$x = x_1^0 \oplus x_1^1 = x_2^0 \oplus x_2^1.$$

Then, for any $\hat{V}$, the induced distribution on $\hat{V}$'s view of

$$\text{PROVE-EQUAL}(x_1^0, x_1^1, x_2^0, x_2^1),$$

followed by the revelation of $x_1^0, x_1^1, x_2^0, x_2^1$, may be generated by the following algorithm.

1. Choose $v \in \{0,1\}$ at random and choose $y_1, y_2 \in \{0,1\}$ uniformly subject to $v = y_1 \oplus y_2$.
2. Send v to $\hat{V}$. On receipt of b from $\hat{V}$, set $x_1^b = y_1$ and $x_2^b = y_2$, and send x_1^b and x_2^b to $\hat{V}$.
3. Set $x_1^{1-b} = x \oplus x_1^b$ and $x_2^{1-b} = x \oplus x_2^b$, and send x_1^0, x_1^1, x_2^0 and x_2^1 to $\hat{V}$.

In particular, Property 2 implies that $\hat{V}$'s view through the PROVE-EQUAL protocol is independent of x.

In the protocol of Figure 3, a pair $(x_i^0[j], x_i^1[j])$ is called *used* if it has been chosen in some previous iteration of the **repeat** loop and *unused* if it has not. Straightforward probabilistic arguments (which we omit) show that the protocol has the following desired properties.

Lemma 5 Suppose that $Q(x_1, \ldots, x_m)$ is false and that $l > 3k$. Then, during each iteration of the **repeat** loop in the PROVE-MANY protocol, V rejects with probability at least

$$1 - \left(1 - \frac{1}{(2m+8)}\right)^k.$$

```
Protocol PROVE-MANY($x_1, \ldots, x_m, Q, k$)
I: repeat $k$ times
    1: $V$ chooses $j_1, \ldots, j_m$ such that, for $1 \leq d \leq m$, $(x_i^0[j_d], x_i^1[j_d])$ is unused,
       and $b \in \{0,1\}$ uniformly.
    2: If $b = 1$, then $V$ and $P$ run PROVE($x_1, \ldots, x_m, Q$), where the pair
       $(x_i^0[j_i], x_i^1[j_i])$ is used to represent $x_i$.
    3: If $b = 0$, then $V$ chooses $j'_1, \ldots, j'_m$ such that $(x_i^0[j'_d], x_i^1[j'_d])$ is unused for
       $1 \leq d \leq m$. Then, for each $d$, $P$ and $V$ run
           PROVE-EQUAL $\left(x_i^0[j_i], x_i^1[j_i], x_i^0[j'_d], x_i^1[j'_d]\right)$.
II: $V$ rejects iff $V$ ever rejected during Steps 2 or 3 of the loop. $P$ aborts the
protocol if ever asked to "reuse" a pair.
```

Figure 3: Protocol for decreasing the probability of error.

Lemma 6 Suppose that $Q(x_1, \ldots, x_m) = Q(\bar{x}_1, \ldots, \bar{x}_m) = 1$, and that, for some set $T \subseteq \{1, \ldots, m\}$, $x_t = \bar{x}_t$ for $t \in T$. Let $\hat{V}$ be an arbitrary computationally unbounded party. Then the distribution on $\hat{V}$'s view induced by running COMMIT$(x_1, \ldots, x_m)$, PROVE-MANY$(x_1, \ldots, x_m, Q, k)$, and REVEAL(t) for $t \in T$ is identical to that induced by running COMMIT$(\bar{x}_1, \ldots, \bar{x}_m)$, PROVE-MANY$(\bar{x}_1, \ldots, \bar{x}_m, Q, k)$, and REVEAL$(t)$ for $t \in T$.

Together Lemmas 2 through 6 give the following:

Theorem 2 Every predicate $Q(x_1, \ldots, x_m)$ has a 1-round, 2^{-m}-error zero-knowledge ICS proof system.

5 Open Questions

Open questions abound, including:
Question 1: Can Theorem 1 be improved so that fewer than $t\lfloor m/c \lg m \rfloor$ random instances are needed? Alternatively, can a lower bound on the required number of random instances be proven?

Currently, it is not even known whether there is a function f that is not $(1,2)$-locally random reducible to any function g. Fortnow and Szegedy [10] show that there is an f that is not $(1,2)$-locally random reducible to a pair of functions (g_1, g_2), if one insists that the functions g_i be boolean and that the reduction have 0 error probability.

Question 2: Is there a protocol for zero-knowledge proofs of arbitrary predicates on committed bits that is even more communication-efficient than the one we have presented?

Question 3: Is there a fixed polynomial m^c with the following property: For any polynomial-time predicate $Q(x_1, \ldots, x_m)$, there is a zero-knowledge protocol that proves the value of Q on committed bits, has bit complexity m^c, and has a prover and verifier that both run in polynomial time? That is, if we restrict attention to poly-time Q's, is there a protocol that shares with the protocol presented in this paper the property that the (polynomial) communication complexity does not depend on the computational complexity of Q and has the additional property that the prover and verifier are poly-time?

6 Acknowledgements

We are grateful to Yvo Desmedt for interesting discussions of these results and to the referees for helpful comments on the presentation.

References

[1] M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle, *J. Comput. System Sci.* **39** (1989), 21–50.

[2] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Proofs, *Computational Complexity* **1** (1991), 3–40.

[3] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries, *Proc. 7th Annual Symposium On Theoretical Aspects Of Computer Science*, Lecture Notes in Computer Science, vol. 415, Springer, Berlin, 1990, 37–48.

[4] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Cryptographic Applications of Locally Random Reductions, *AT&T Bell Laboratories Technical Memorandum*, November 15, 1989.

[5] C. Bennett. Private communication via Gilles Brassard.

[6] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastaad, J. Kilian, S. Micali, and P. Rogaway. Everything Provable is Provable in Zero-Knowledge, *Advances in Cryptology – Crypto '88*, Lecture Notes in Computer Science, vol. 403, Springer, Berlin, 1990, 37–56.

[7] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, *Proc. 20th Annual Symposium on Theory of of Computing,*, ACM Press, New York, 1988, 1–10.

[8] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Proofs, *Proc. 22nd Annual Symposium on Theory of Computing*, ACM Press, New York, 1990, 416–426.

[9] J. Feigenbaum. Locally Random Reductions in Interactive Complexity Theory, *Advances in Computational Complexity*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, AMS, Providence, 1993, 73–98.

[10] L. Fortnow and M. Szegedy. On the Power of Two-Oracle Instance-Hiding Schemes, *Inform. Proc. Ltrs.* **44** (1992), 303–306.

[11] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design, *J. ACM* **38** (1991), 691–729.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game, *Proc. 19th Annual Symposium on Theory of Computing*, ACM Press, New York, 1987, 218–229.

[13] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems, *SIAM J. Comput.* **18** (1989), 186–208.

[14] R. Impagliazzo and M. Yung. Direct Minimum Knowledge Computations, *Advances in Cryptology – Crypto '87*, Lecture Notes in Computer Science, vol. 293, Springer, Berlin, 1988, 40–51.

[15] R. Lipton. New Directions in Testing, in *Distributed Computing and Cryptography*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 2, American Mathematical Society, Providence, 1991, 191–202.

[16] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems, *J. ACM* **39** (1992), 859–868.

[17] R. Rivest. Workshop on Communication and Computing, MIT, October 1986.

[18] S. Rudich. Private communication via Gilles Brassard.

[19] A. Shamir. IP = PSPACE, *J. ACM* **39** (1992), 869–877.