Security with Low Communication Overhead

D. Beaver* J. Feigenbaum† J. Kilian‡ P. Rogaway§

(Extended Abstract)

Abstract

We consider the communication complexity of secure multiparty computations by networks of processors each with unlimited computing power. Say that an n-party protocol for a function of m bits is efficient if it uses a constant number of rounds of communication and a total number of message bits that is polynomial in $\max(m, n)$. We show that *any* function has an efficient protocol that achieves $(n \log n)/m$ resilience. Ours is the first secure multiparty protocol in which the communication complexity is independent of the computational complexity of the function being computed.

We also consider the communication complexity of zero-knowledge proofs of properties of committed bits. We show that every function f of m bits has an efficient *notarized envelope scheme*; that is, there is a protocol in which a computationally unlimited prover commits a sequence of bits x to a computationally unlimited verifier and then proves in perfect zero-knowledge (without decommitting x) that $f(x) = 1$, using a constant number of rounds and $poly(m)$ message bits. Ours is the first notarized envelope scheme in which the communication complexity is independent of the computational complexity of f.

Finally, we establish a new upper bound on the number of oracles needed in *instance-hiding schemes* for arbitrary functions. These schemes allow a computationally limited querier to capitalize on the superior power of one or more computationally unlimited oracles in order to obtain $f(x)$ without revealing its private input x to any one of the oracles. We show that every function of m bits has an $(m/\log m)$-oracle instance-hiding scheme.

The central technique used in all of these results is *locally random reducibility*, which was used for the first time in [7] and is formally defined for the first time here. In addition to the applications that we present, locally random reducibility has been applied to interactive proof systems, program checking, and program testing.

*AT&T Bell Laboratories, Room 2C324, 600 Mountain Avenue, Murray Hill, NJ 07974 USA, beaver@research.att.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 USA, jf@research.att.com.

‡Harvard University and MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139 USA, joek@theory.lcs.mit.edu. Supported by an NSF Postdoctoral Fellowship.

§MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139 USA, rogaway@theory.lcs.mit.edu.

A.J. Menezes and S.A. Vanstone (Eds.): Advances in Cryptology - CRYPTO '90, LNCS 537, pp. 62-76, 1991.

1 Introduction

A *resilient, multiparty protocol* allows a network of processors to compute a function of the processors' private inputs in such a way that each processor learns the result, no processor learns anything about another's private input (except what is implied by the result and its own private input), and the protocol works even if some proper subset of the processors behave in an arbitrary faulty manner. For example, a t-resilient protocol for majority-voting allows all voters to learn who won, while preventing any coalition of t or fewer anarchists from learning the vote of an honest participant or disrupting the election.

In this paper, we make no assumptions about the computing power of the individual processors in our network. Because no such assumptions are made, all proofs of the security properties of protocols must be argued on information-theoretic grounds. No complexity-theoretic hypotheses, such as the existence of one-way functions, are relevant.

Let n be the number of processors and m be the total number of input bits to the function being computed. We focus on three criteria for evaluating a protocol:

- The *resilience* $t(m, n)$ is the maximum number of faulty processors that can be tolerated.
- The *round complexity* $\rho(m, n)$ is the maximum number of communication rounds in an execution of the protocol.
- The *bit complexity* $b(m, n)$ is the maximum total number of message bits sent by *all* of the processors during an execution.

Theorem: For any positive constant c, every function of m bits has an n-party protocol that achieves $(cn \log n)/m$ resilience, *constant* round complexity, and bit complexity that is *polynomial* in $\max(m, n)$.

That is, even functions whose circuit complexity is exponential have resilient multiparty protocols whose communication complexity is polynomial.

Our actual result gives a more general tradeoff and is stated precisely in Section 5.3 below.

In a resilient multiparty protocol, the processors are equally powerful, and they must cooperate in a computation because none of them alone has all of the necessary data. An alternative view of distributed computations with private data is considered in [1, 7]. There one processor, who has all of the data, must cooperate with other processors because it lacks the power to carry out the computation. More specifically, a *μ-oracle instance-hiding scheme* for a function f is a protocol in which a polynomial-time bounded *querier* consults μ computationally-unlimited *oracles* in such a way that the querier learns the value $f(x)$, but none of the oracles learns the input x. Beaver and Feigenbaum [7] show that, for every positive constant c, every function of m bits has an $(m - c \log m)$-oracle instance-hiding scheme. In this paper, we improve this general upper bound.

Theorem: For every positive constant c, every function of m bits has an $(m/c \log m)$-oracle instance-hiding scheme.

Zero-knowledge proof systems, as originally formulated in [25], 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 a proof. *Notarized envelope schemes* allow the prover to publish a *commitment* to its private input x and then prove in zero-knowledge to the verifier that $f(x) = 1$ *without decommitting* x. Many of the cost criteria that apply to resilient multiparty protocols also apply to notarized envelope schemes. In this paper, we are most interested in the communcation costs of a scheme – that is, its round complexity and its bit complexity.

We examine notarized envelope schemes in the *ideal envelope model*: that is, both prover and verifier have unlimited computational power, no cryptographic assumptions are made, and bit commitment is assumed as a primitive. Any primitive that implements bit commitment in this model is called an *envelope scheme*. A natural question to ask is whether notarized envelope schemes exist in this model; that is, can notarized envelopes be built out of ordinary envelopes? This question was answered in the affirmative by numerous authors (e.g., [10, 32]); a written account of one scheme appears in [11].

All previously published notarized envelope schemes have the following feature in common: They have bit complexity proportional to the circuit complexity of f. Here, we achieve a more communication-efficient construction of general notarized envelope schemes.

Theorem: In the ideal envelope model, every function has a notarized envelope scheme that has *constant* round complexity and bit complexity *polynomial in the number of input bits.*

The results given here first appeared in our Technical Memorandum [8]. In this abstract, some details of proof are omitted, because of space limitations; many of these details can be found in [6], and all will appear in our final paper.

The rest of this abstract is organized as follows. Section 2 contains a brief account of previous work on secure, distributed computation, with emphasis on the results that do not involve complexity-theoretic hypotheses. Notation and terminology is given in Section 3, and several necessary building blocks are recalled from the literature. Section 4 gives a precise definition of locally random reducibility and recalls the reduction given in [7]. We present our main results in Section 5; the multiparty protocol result is given last, because it relies upon the other two.

2 Previous Work on Secure Distributed Computation

Secure distributed computation was introduced by Yao [35]. His proofs of the security properties of protocols are based on complexity-theoretic hypotheses such as

the existence of one-way functions. Further work along these lines can be found in [9, 18, 23, 24, 36].

Previous results on secure distributed computation without complexity-theoretic hypotheses can be summarized as follows.

- The approach was first taken by Ben-Or, Goldwasser, Wigderson [13] and Chaum, Crépeau, Damgård [17]. Their n-party protocols for secure evaluation of a function f of m bits are based on distributed simulation of a circuit C_f for f. Their protocols achieve $(n-1)/3$-resilience, round complexity proportional to the depth of C_f, and bit complexity proportional to the size of C_f.
- T. Rabin, Ben-Or [31] and Beaver [4] give protocols that achieve $(n-1)/2$-resilience but have the same round complexity and the same bit complexity as those in [13, 17].
- The protocol of Bar-Ilan, Beaver [3] achieves $(n-1)/2$-resilience, *constant* round complexity, and bit complexity proportional to the size of C_f, if the depth of C_f is $O(\log m)$.
- Chaum [16] gives an interesting protocol that can be proven $(n-1)/2$-resilient without complexity-theoretic hypotheses and $(n-1)$-resilient with complexity-theoretic hypotheses.

Throughout this paper, we use the term "secure protocol" to mean a protocol that is "resilient" against arbitrarily (i.e., potentially malicious) faulty players. There is also a literature on a weaker notion of security – "privacy" against "honest but curious" players; refer to [5, 19, 27] for details.

3 Preliminaries

We use f to denote a function with domain $\{0,1\}^m$. The range of f is contained in $K_{m,n}$ – a finite field that is large enough but still of size polynomial in $\max(m,n)$. The meaning of "large enough" will vary but will be clear from context. The field $K_{m,n}$ will always be constructible in time polynomial in $\max(m,n)$. The constants $\alpha_1, \alpha_2, \ldots$ are distinct elements of $K_{m,n} \setminus \{0\}$.

Consider g, the "arithmetization" of f over $K_{m,n}$. For $A = (a_1, \ldots, a_m) \in \{0,1\}^m$, let

$$\delta_A(X_1, \ldots, X_m) = \prod_{j=1}^{m} (X_j - \overline{a_j})(-1)^{\overline{a_j}} \in K_{m,n}[X_1, \ldots, X_m].$$

Then, for any $x = (x_1, \ldots, x_m) \in \{0,1\}^m$, $\delta_A(x)$ is 1 if $A = x$ and it is 0 otherwise. Next, let

$$g(X_1, \ldots, X_m) = \sum_{A \in \{0,1\}^m} f(A)\delta_A(X_1, \ldots, X_m) \in K_{m,n}[X_1, \ldots, X_m].$$

The multivariate polynomial g has the property that it agrees with f on all inputs in $\{0,1\}^m$.

For example, if f is the boolean function

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3,$$

where $\oplus$ denotes exclusive-or, the arithmetization is

$$g(X_1, X_2, X_3) = X_1(1-X_2)(1-X_3)+(1-X_1)X_2(1-X_3)+(1-X_1)(1-X_2)X_3+X_1X_2X_3.$$

Our model of computation is a synchronous network of computationally unlimited processors, P_1, ..., P_n, completely connected by private channels and a broadcast channel, acted upon by a dynamic adversary. In one round of the protocol, each processor can receive a message from each other processor, perform an unlimited amount of local computation, and send a message to each other processor. Each processor P_i has a private input z_i and a private random source. The correct outcome of the protocol is, roughly, that each P_i receives the functional value while no P_i receives any information about z_j, $j \neq i$, except what is implied by the functional value and z_i. Furthermore, the malicious players' inputs do not depend on the honest players' inputs. The protocol is *t-resilient* if the correct outcome occurs even if as many as t of the processors exhibit arbitrarily faulty behavior.

The vector $(z_1, \ldots, z_n)$ of inputs contains a total of m bits, which we denote by x_1, ..., x_m. Thus the desired functional value is $f(x_1, \ldots, x_m)$.

An *envelope scheme* is a protocol in which two *computationally-unlimited* processors achieve *information-theoretically secure bit commitment.* That is, *committer* P_1 can compute and send to the *receiver* P_2 a sequence of *commitments*, say c_1, ..., c_m, to a sequence of bits b_1, ..., b_m. The sequence c_1, ..., c_m conveys no information (in the Shannon sense) to the receiver about the sequence b_1, ..., b_m (except its length). At any time after the commitment takes place, P_2 may challenge P_1 to *decommit* any c_i. The envelope scheme must have the properties that P_1 can always prove that c_i is a commitment to b_i and that P_1 can never prove that c_i is a commitment to $1 - b_i$. Notice that, by definition, the security of an envelope scheme is two-sided: P_2 cannot find out any information about b_i unless P_1 decommits, and P_1 cannot "change its mind" about what it has committed.

In Section 5.2 below, we present a result in the *ideal envelope model*, as opposed to the cryptographic model. That is, we assume that committer and receiver both have unlimited computational power and that an envelope scheme is given as a primitive.

A *notarized envelope scheme* is a protocol in which a *prover* commits to a sequence of bits x_1, ..., x_m, and then gives a *zero-knowledge proof* to a *verifier* that $f(x_1, \ldots, x_m) = 1$, *without revealing any information* about x_1, ..., x_m. Since the verifier has unlimited computational power in the ideal envelope model, a natural question to ask is why it cannot simply compute $f(x_1, \ldots, x_m)$ without the help of the prover. The answer, of course, is that the verifier does not have x_1, ..., x_m; rather it has the prover's commitments to x_1, ..., x_m.

The round complexity of a notarized envelope scheme or n-party protocol for a function f of m bits is the (worst-case) total number of rounds in an execution, as

a function of n and m. Similarly, the bit complexity is the sum, over all i, of the sum of the lengths of all messages sent by P_i in a (worst-case) execution. We use the following results in our constructions in Section 5.

Theorem 3.1 ([10, 11, 32]) *If f is computable by a (possibly nondeterministic) boolean or arithmetic circuit of polynomial size, then it has a notarized envelope scheme with constant round complexity and bit complexity polynomial in the length of the input.*

Theorem 3.2 ([3]) *If f is computable by a boolean or arithmetic circuit of depth $\log m$ and size poly(m), then f has an $((n-1)/2)$-resilient, n-party protocol with constant round complexity and bit complexity poly$(\max(m,n))$.*

In Section 5, we refer to the constant-round, polynomial-bit notarized envelope schemes of [10, 11, 32] for functions with poly-size circuits as the *standard notarized envelope schemes.*

A *$\mu(m)$-oracle instance-hiding scheme for f* is a $\rho(m)$-round synchronous protocol executed by a polynomial-time Turing Machine P_0 (the *querier*) and $\mu(m)$ computationally-unlimited oracle Turing Machines $P_1, \ldots, P_\mu$ (the *oracles*). Querier P_0 has a private input $x \in \{0,1\}^m$ as well as a private source of randomness. For each oracle P_i, there is a private channel connecting P_0 to P_i. There are no communication channels at all between pairs of distinct oracles. This model is analogous in one sense to the multiprover model of interactive proofs systems (see [2, 12, 22]): The oracles (like the provers) can "meet to agree on a strategy" before the execution of the instance-hiding scheme begins, but they cannot collude during the execution of the scheme, and no oracle sees the communication between the querier and the other oracles. During one round of the scheme, P_0 can receive a message from each oracle, perform a randomized polynomial-time local computation, and send a message to each oracle. Also in one round, each oracle can receive a message from P_0, perform an unlimited amount of local computation, and send a message to P_0. In round $\rho(m)$, P_0 uses the transcript of the execution to compute $f(x)$. For $1 \leq i \leq \mu(m)$, the sequence of messages sent by P_0 to P_i is independent of x, given m.[1] Intuitively, P_0 uses the oracles to compute $f(x)$ without telling any one of them what x is.

4 Locally Random Reductions

Intuitively, a *locally random reduction* from a function f to a function g is a randomized polynomial-time mapping that takes an *arbitrary* instance x of f to a set $\{y_1, \ldots, y_\mu\}$ of *random* instances of g in such a way that $f(x)$ is easily computable from $g(y_1), \ldots, g(y_\mu)$. A function f is *random self-reducible* if there is a locally random reduction from f to itself. We now define this concept more formally.

[1] There is actually no need to restrict attention to instance-hiding schemes that "leak at most n." Refer to [7, Section 2] for a detailed description of the general model.

Definition 4.1 *A function f is* **$(\mathrm{t}(\mathrm{m}), \mu(\mathrm{m}))$-locally random reducible** *to a function g if there are polynomial-time computable functions ϕ, σ_1, ..., $\sigma_{\mu(m)}$ and a polynomially-bounded function $w(m)$ with the following properties.*

- *For all m, all $x \in \{0,1\}^m$, and at least $3/4$ of all $r \in \{0,1\}^{w(m)}$,*

$$f(x) = \phi(x, r, g(\sigma_1(x,r)), \ldots, g(\sigma_{\mu(m)}(x,r))).$$

- *If r is chosen uniformly from $\{0,1\}^{w(m)}$, then, for any pair x_1, x_2 of distinct elements of $\{0,1\}^m$ and all $\{i_1, \ldots, i_{t(m)}\} \subset \{1, \ldots, \mu(m)\}$, the random variables $\langle \sigma_{i_1}(x_1,r), \ldots, \sigma_{i_{t(m)}}(x_1,r)\rangle$ and $\langle \sigma_{i_1}(x_2,r), \ldots, \sigma_{i_{t(m)}}(x_2,r)\rangle$ are identically distributed.*

More succinctly, we say that f is (t,μ)-lrr to g.

Informally, if T is a subset of the target instances $\{y_1 = \sigma_1(x,r), \ldots, y_{\mu(m)} = \sigma_{\mu(m)}(x,r)\}$, and $|T| \leq t(m)$, then T leaks no information about the original instance x, except its length m.

As in the definition of BPP, the fraction $3/4$ can be replaced by $1/2+1/poly(m)$. In the results we present below, the equation $f(x) = \phi(x, r, g(\sigma_1(x,r)), \ldots, g(\sigma_{\mu(m)}(x,r)))$ can be made to hold for all r, by appropriate choice of the field $K_{m,n}$.

Note that a $(1,\mu)$-lrr from f to g can be thought of as a one-round, μ-oracle instance-hiding scheme for f. The querier P_0 chooses r, computes the target instances $y_i = \sigma_i(x,r)$, for $1 \leq i \leq \mu$, and sends each y_i to a separate g-oracle P_i. P_i then sends back $g(y_i)$, for $1 \leq i \leq \mu$, and P_0 computes $f(x)$ using ϕ, x, r, and the values $\{g(y_i)\}$.

The following special case of Definition 4.1 is important: f is $(1,\mu)$-lrr to g, and each of the random instances $\sigma_i(x,r)$ is distributed uniformly over all g-instances of the appropriate length (although, as usual, pairs of instances $\sigma_i(x,r)$ and $\sigma_j(x,r)$ are dependent). In this case, the average-case complexity of g gives an upper bound (up to polynomial factors) on the worst-case complexity of f. This property of locally random reductions is used to prove that the permanent function is as hard on average as it is in the worst case (cf. [28]) and to show that, for every (finite) function f, there is a polynomial g such that g is as hard on average as f is in the worst case (cf. [7]).

More precisely, it is shown in [7] that every function f of m inputs bits is $(1, m+1)$-lrr to a polynomial g. We repeat this construction here and improve upon it in Section 5.1 below.

Let $x = (x_1, \ldots, x_m)$ be an element of $\{0,1\}^m$. Choose m coefficients c_1, ..., c_m independently and uniformly from $K_{m,n}$. The univariate polynomial

$$G(Z) = g(c_1 Z + x_1, \ldots, c_m Z + x_m)$$

is of degree m and has the property that

$$G(0) = g(x_1, \ldots, x_m) = f(x_1, \ldots, x_m).$$

For $1 \leq i \leq m+1$, the function σ_i of the locally random reduction maps x to $y_i = (c_1\alpha_i + x_1, \ldots, c_m\alpha_i + x_m)$. The function ϕ recovers $f(x)$ by interpolating the

pairs $(\alpha_1, g(y_1)), \ldots, (\alpha_{m+1}, g(y_{m+1}))$; these $m+1$ points determine a unique degree-m polynomial in $K_{m,n}[Z]$ - namely $G(Z)$, which has constant term $f(x)$. Finally, each y_i is distributed uniformly over $K_{m,n}^m$ and hence leaks no information about x.

All of the results we present in Section 5 use locally random reductions in an essential way. This notion of reducibility has also been applied to interactive proof systems (cf. [2, 29, 34]), and to program checking, testing, and self correcting (cf. [14, 15, 28]). An extensive treatment of the complexity-theoretic aspects of random self-reducibility can be found in [20, 21].

5 Results

5.1 Instance-Hiding Schemes

Theorem 5.1 *For every function f and every polynomial $t(m)$, there is a polynomial h such that f is $(t(m), 1 + t(m)m/\log m)$-lrr to h.*

Proof (sketch): Clearly, it suffices to prove the following two lemmas.

Lemma 5.1 *Every function f is polynomial-time, many-one reducible to a multivariate polynomial h over $K_{m,n}$ of degree $m/\log m$.*

Lemma 5.2 *For every polynomial $t(m)$, every multivariate polynomial h over $K_{m,n}$ is $(t, 1 + dt)$-locally random* **self-reducible**, *where d is the degree of h.*

Let f be a function that maps $\{0,1\}^m$ to $K_{m,n}$. Assume without loss of generality that $\log m$ divides m; if it does not, then x can be "padded" with dummy input bits. Divide the input bits $x_1, \ldots, x_m$ into consecutive "blocks" of length $\log m$. For example, if $m = 4$, then the first block is $\{x_1, x_2\}$ and the second is $\{x_3, x_4\}$.

Proof of Lemma 5.1: The crux of this construction is a change of variables that allows us to use a degree-$(m/\log m)$ polynomial h in $m^2/\log m$ variables in place of the degree-m polynomial g in m variables that was used in Section 4.

Let $\{x_j\}_{j=(k-1)\log m+1}^{k\log m}$, for each $1 \leq k \leq m/\log m$, be a block of input bits, and let $\{X_j\}_{j=(k-1)\log m+1}^{k\log m}$ be the corresponding block of indeterminates over $K_{m,n}$ that are used to define the arithmetization g of f. For each subset S of the indices $\{(k-1)\log m + 1, \ldots, k\log m\}$, let the variable W_S represent the monomial $\Pi_{j\in S} X_j$. There are $m/\log m$ blocks, and hence a total of $m^2/\log m$ variables $W_1, \ldots, W_{m^2/\log m}$.

Each monomial in g can be represented as a monomial in $m/\log m$ of the W's. Let $h(W_1, \ldots, W_{m^2/\log m})$ be the degree-$(m/\log m)$ polynomial that results from summing these representations of all of the monomials in g. ∎

Proof of Lemma 5.2: Let $h(W_1, \ldots, W_s)$ be a degree-d polynomial over $K_{m,n}$, and let $w = (w_1, \ldots, w_s)$ be an element of $K_{m,n}^s$. We show how to reduce w to a collection of $1 + dt$ elements of $K_{m,n}^s$ with an appropriate distribution. For each $1 \leq l \leq s$, choose t elements $c_{l,1}, \ldots, c_{l,t}$ of $K_{m,n}$ independently and uniformly at random, and let $q_l(Z) = c_{l,t}Z^t + \cdots + c_{l,1}Z + w_l$. Then the univariate polynomial

$$H(Z) = h(q_1(Z), \ldots, q_s(Z))$$

is of degree at most $t(m) \cdot d$ and has the property that

$$H(0) = h(w_1, \ldots, w_s).$$

For $1 \leq i \leq 1 + td$, the function $\sigma_i(x, r)$ maps x to $y_i = (q_1(\alpha_i), \ldots, q_s(\alpha_i))$. The function ϕ recovers $h(w)$ by interpolating the polynomial $H(Z)$ from the pairs $(\alpha_i, h(y_i))$.

The second requirement of Definition 4.1 is satisfied for the same reason that Shamir's secret-sharing scheme works (cf. [33]): For every l, t or fewer values of the random degree-t polynomial q_l reveal no information about its constant term w_l. ∎

The relationship between the multivariate polynomial g and the univariate polynomial G in Section 4 is a special case of the relationship between the polynomials h and H that we use here; in that special case, $t = 1$. ∎

Remark 5.1 *This can be improved to a $(t(m), 1+t(m)m/c\log m)$-lrr, for any positive constant c, by increasing the block size to $c \log m$ (and the number of w_l's per block to m^c).*

Remark 5.2 *More generally, one can use block size k, which is not necessarily $O(\log m)$. This will result in a $(t(m), 1 + t(m) \cdot \lceil m/k \rceil)$-lrr from f to a polynomial h of degree $\lceil m/k \rceil$ in $2^k \cdot \lceil m/k \rceil$ variables, but the reduction will take time $2^k \cdot poly(m)$.*

By taking $t(m) = 1$, we get the desired result on instance-hiding schemes.

Corollary 5.1 *For every positive constant c, every function has an $(m/c\log m)$-oracle instance-hiding scheme.*

5.2 Notarized Envelope Schemes

In this section, we show how to build a notarized envelope scheme with low communication complexity, starting with an ordinary envelope scheme. The notation f and g is as in Section 3.

Let P and V denote the prover and verifier of the notarized envelope scheme. In most of the literature on interactive proof systems, V denotes a probabilistic, polynomial-time verifier. Therefore, we stress that, in a notarized envelope scheme, *no limitation is placed of the computational power of V.*

Intuitively, player P in our notarized envelope scheme plays the roles of the querier and all of the oracles in the instance-hiding scheme of [7]. Player V then challenges the prover to demonstrate that it played both roles faithfully. In the following protocol, the quantifiers "for $1 \leq i \leq m + 1$" and "for $1 \leq j \leq m$" are implicit in each step in which the subscript i or j occurs.

Notarized Envelope Scheme to show that $f(x_1, \ldots, x_m) = 1$

Step 1. P commits to x_j.

Step 2. P selects random c_j uniformly at random from $K_{m,n}$ and lets $q_j(Z) = c_j Z + x_j$. Then P computes $(y_{1,i}, \ldots, y_{m,i}) = (q_1(\alpha_i), \ldots, q_m(\alpha_i))$ and $v_i = g(y_{1,i}, \ldots, y_{m,i})$.

Step 3. P commits to q_j, $(y_{1,i}, \ldots, y_{m,i})$, and v_i.

Step 4. P uses a standard notarized envelope scheme to prove to V that $q_j(0) = x_j$, that $q_j(\alpha_i) = y_{j,i}$, and that $\{(\alpha_i, v_i)\}$ interpolate a degree-m polynomial with constant term 1. V rejects (and terminates the protocol) if any of these proofs fail.

Step 5. V chooses k uniformly from $\{1, \ldots, m+1\}$ and sends it to P.

Step 6. P decommits $(y_{1,k}, \ldots, y_{m,k})$, and v_k.

Step 7. V accepts if and only if $g(y_{1,k}, \ldots, y_{m,k}) = v_k$.

Theorem 5.2 *The protocol just given is a notarized envelope scheme and has the following properties.*

(A) *If $f(x_1, \ldots, x_m) = 1$ and P is honest, then V always accepts.*

(B) *If $f(x_1, \ldots, x_m) \neq 1$ and P cheats, then V rejects with probability at least $1/(m+1)$.*

(C) *Repetitions can be performed in parallel. Thus the rejection probability in* (B) *can be amplified while retaining constant round complexity and polynomial bit complexity.*

Proof (sketch): The fact that this protocol satisfies the requirements of a notarized envelope scheme follows from the fact that $q_j(\alpha_i)$ is uniformly distributed over $K_{m,n}$, the properties of the standard notarized envelope schemes, the definition of envelopes, and the fact that V has the power to compute g. We first argue that this protocol satisfies the necessary security requirements. That is, we claim that no information about $x_1, \ldots, x_m$ (but for the value of m) is revealed to V. By the properties of the standard notarized envelope schemes, the bits revealed during any of these zero-knowledge proofs are independent of the values of $x_1, \ldots, x_m$. It also follows from a straightforward analysis that these revealed bits are independent of the values of $y_{1,k}, \ldots, y_{m,k}$ for $1 \leq k \leq m+1$. By the properties of our locally random reduction, we have that $y_{1,k}, \ldots, y_{m,k}$ is uniformly distributed over K_m^m, and is thus independent of $x_1, \ldots, x_m$. Finally, we note that v_k is completely determined by $y_{1,k}, \ldots, y_{m,k}$, and thus contributes no additional information about $x_1, \ldots, x_m$.

Part (A) follows from the properties of the older notarized envelope schemes, and the fact that the prover never makes an untrue assertion.

To prove (B), observe that, if P cheats, then either one of the assertions that he makes in Step 4 is wrong (and thus V will reject with high probability), or at least one of the v_i's is not equal to $g(y_{1,i}, \ldots, y_{m,i})$ (and thus V will reject with probability at least $1/(m+1)$ in Step 7).

Part (C) follows from the parallelizability of the earlier notarized envelope schemes. Note that we are working in the ideal envelope model, in which zero-knowledge proofs can be run in parallel without losing their zero-knowledge properties. ∎

5.3 Resilient Multiparty Protocols

Theorem 5.3 *For any positive constant c, every function f has a $((cn \log n)/m)$-resilient, n-party protocol with constant round complexity and bit complexity polynomial in* $\max(m, n)$.

Proof (sketch): We make use of a $((cn \log n)/m, n)$-lrr from f to a polynomial h of degree $m/c' \log n$, where $c' > c$. Such a reduction is obtained by taking block size $c' \log n$ in Remark 5.2. Let $s = n^{c'} m/c' \log n$ be the number of variables w_l in h. Recall that player P_i's input is z_i, which is some subset of the bits $x_1, \ldots, x_m$.

We first exhibit a protocol that satisfies the weaker requirement of *t-privacy*, where $t = ((cn \log n)/m)$. That is, as many as t players may collude to discover others' inputs, but all players compute and send all required values correctly. We then show how to enhance the basic protocol to achieve t-resilience. When we "run a subprotocol" to compute intermediate values, we use the resilient n-party protocol of Bar-Ilan and Beaver [3].

We say that a player *t-secretly shares b* when the player selects $q(Z) = c_t Z^t + \cdots + c_1 Z + b$ by choosing each c_i independently and uniformly at random, and sends $q(\alpha_i)$ to P_i, for $1 \leq i \leq n$. The quantifiers "for $1 \leq i, j \leq n$" and "for $1 \leq l \leq s$" are implicit in each step in which the subscripts i, j, and l occur.

An n-party protocol to compute $f(x_1, \ldots, x_m)$

Step 1. P_i t-secretly shares each bit of z_i.

Step 2. Run a subprotocol to compute w_l and t-secretly share it. Let $(y_{1,i}, \ldots, y_{s,i})$ denote the shares sent to P_i.

Step 3. P_i computes $v_i = h(y_{1,i}, \ldots, y_{s,i})$ and t-secretly shares it.

Step 4. Run a subprotocol to interpolate the polynomial $H(Z)$ from $\{(\alpha_i, v_i)\}$ and reveal to everyone the constant term, which is $f(x_1, \ldots, x_m)$.

Observe first that this basic protocol is correct, is t-private, has constant round complexity, and has polynomial bit complexity. Essentially, this follows from the properties of Shamir's secret-sharing scheme and from Theorem 3.2, because the subprotocols of Steps 2 and 4 use poly-size, log-depth circuits: change of variables, selection of random polynomials, polynomial evaluation and polynomial interpolation.

To achieve $((cn \log n)/m)$ *resilience*, we add another subprotocol between Steps 3 and 4. In this subprotocol, each player P_i proves in zero-knowledge that he has computed v_i correctly. These proofs are accomplished using the notarized envelope scheme of Section 5.2 and a (constant-round, polynomial-bits) majority-voting scheme. "Envelopes" need *not* be assumed as a primitive (as they are in the ideal envelope model of Section 5.2), because they can be implemented using verifiable secret-sharing (cf. T. Rabin [30]). In the resilient version of the protocol, verifiable secret sharing is used in all steps in which ordinary secret sharing is used in the private version. ∎

Remark 5.3 *The bound $t = (cn \log n)/m$ is needed because h has degree $d = m/c' \log n$, and we need $dt < n$ in order to interpolate the result. That is, even if the processors are "honest but curious," our protocol is only $((cn \log n)/m)$-private, because it must compute the locally random reduction. Everything except the locally random reduction could be made $(n-1)/2$ resilient. In general, any t-private, n-party protocol for f with constant round complexity and bit complexity polynomial in $\max(m, n)$ can be compiled into a t-resilient, n-party protocol for f with constant round complexity and bit complexity polynomial in $\max(m, n)$.*

Correctness proofs for multiparty protocols are both complicated and elusive. To our knowledge, no proof of correctness for any general multiparty protocol has been widely examined and found to be completely rigorous. Indeed, there is no universally accepted standard for what secure multiparty computation should actually entail. Furthermore, there are no rigorously proven composition results for these protocols. Any rigorous use of these basic protocol constructions must contend with these difficulties.

It is likely that the techniques necessary to prove the results of [3, 13, 17, 31] cleanly and rigorously will also suffice to analyze our protocol. In the meantime, our strategy is to compartmentalize these earlier results so that we can truly treat them as black boxes. Toward this end, we will first consider a *trusted servant* abstraction for multiparty protocols. In addition to the n players, of potentially infinite power, we include an auxiliary player (the servant) with the following properties.

1. The servant is guaranteed to behave honestly.
2. The servant can communicate privately with any of the other players.
3. The servant can perform polynomial time computations on its private data.
4. The servant can terminate without revealing any of its private information.

Our proof can then be divided up as follows. First, and easiest, we restate our protocol so that it uses a trusted servant and prove its correctness in that model. Then we reduce the correctness of the trusted servant protocol to the correctness of results that have already appeared in the literature. At this point, a rigorous proof of the earlier multiparty results will yield a rigorous proof of our result as well.

6 Open Problems

The obvious question is whether the bounds achieved here can be improved. Specifically:

1. Is there a better general upper bound than $m/\log m$ oracles for instance-hiding?
2. Can more than $n \log n/m$ resilience be achieved, while retaining constant round complexity and polynomial bit complexity?

3. If the answer to Question 2 is no for general f, what can be said about functions with polynomial-sized circuits?

4. Because it uses the arithmetization of f, our multiparty protocol incurs a high local computation cost, even if f has small circuit complexity. Can this be avoided?

7 Acknowledgements

The authors are grateful to Yvo Desmedt for interesting discussions of these results and to the organizers of the October, 1989 *Monte Verita Seminar on Future Directions in Cryptography*, where some of this work was done.

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, *Proc. of FOCS 1990*, IEEE.

[3] J. Bar-Ilan and D. Beaver. Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds, *Proc. of PODC 1989*, ACM, 201–209.

[4] D. Beaver. Secure Multiparty Protocols Tolerating Half Faulty Processors, to appear in *J. Cryptology*. Preliminary version in *Proc. of CRYPTO 1989*, Springer Verlag LNCS 435, 560–572.

[5] D. Beaver. Perfect Privacy for Two-Party Protocols, *Proc. of DIMACS Workshop on Distributed Computing and Cryptography* (Princeton, NJ; October, 1989), AMS, 1990.

[6] D. Beaver. Security, Fault-Tolerance, and Communication Complexity for Distributed Systems, PhD Thesis, Harvard University, 1990.

[7] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries, *Proc. of STACS 1990*, Springer Verlag LNCS 415, 37–48.

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

[9] D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols, *Proc. of STOC 1990*, ACM, 503–513.

[10] C. Bennett, G. Brassard, and C. Crépeau. Private communication.

[11] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, J. Kilian, S. Micali, and P. Rogaway. Everything Provable is Provable in Zero-Knowledge, *Proc. of CRYPTO 1988*, Springer Verlag LNCS 403, 37–56.

[12] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability, *Proc. of STOC 1988*, ACM, 113–131.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, *Proc. of STOC 1988*, ACM, 1–10.

[14] M. Blum, M. Luby, and R. Rubinfeld. Stronger Checkers and General Techniques for Numerical Problems, *Proc. of STOC 1990*, ACM, 73–83.

[15] M. Blum, M. Luby, and R. Rubinfeld. Program Result Checking Against Adaptive Programs and in Cryptographic Settings, *Proc. of DIMACS Workshop on Distributed Computing and Cryptography* (Princeton, NJ; October, 1989), AMS, 1990.

[16] D. Chaum. The Spymasters Double-Agent Problem: Multiparty Computations Secure Unconditionally from Minorities and Cryptographically from Majorities, *Proc. of CRYPTO 1989*, Springer Verlag LNCS 435, 591–604.

[17] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols, *Proc. of STOC 1988*, ACM, 11–19.

[18] D. Chaum, I. Damgård, and J. van de Graaf. Multiparty Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output, *Proc. of CRYPTO 1987*, Springer Verlag LNCS 293, 87–119.

[19] B. Chor, E. Kushilevitz. A Zero-One Law for Boolean Privacy, *Proc. of STOC 1989*, ACM, 62–72.

[20] J. Feigenbaum and L. Fortnow. On the Random-Self-Reducibility of Complete Sets, University of Chicago Technical Report 90-22, Computer Science Department, August 20, 1990.

[21] J. Feigenbaum, S. Kannan, and N. Nisan. Lower Bounds on Random-Self-Reducibility, *Proc. of Structures 1990*, IEEE, 100–109.

[22] L. Fortnow, J. Rompel, and M. Sipser. On the Power of Multi-Prover Interactive Protocols, *Proc. of Structures 1988*, IEEE, 156–161.

[23] Z. Galil, S. Haber, and M. Yung. Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model, *Proc. of CRYPTO 1987*, Springer Verlag LNCS 293, 135–155.

[24] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game, *Proc. of STOC 1987*, ACM, 218–229.

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

[26] J. Kilian. Founding Cryptography on Oblivious Transfer, *Proc. of STOC 1988*, ACM, 20–31.

[27] E. Kushilevitz. Privacy and Communication Complexity, *Proc. of FOCS 1989*, IEEE, 416–421.

[28] R. Lipton. New Directions in Testing, *Proc. of DIMACS Workshop on Distributed Computing and Cryptography* (Princeton, NJ; October, 1989), AMS, 1990.

[29] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algrebraic Methods for Interactive Proof Systems, *Proc. of FOCS 1990*, IEEE.

[30] T. Rabin. Robust Sharing of Secrets When the Dealer is Honest or Cheating, M.Sc. Thesis, Hebrew University, 1988.

[31] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority, *Proc. of STOC 1989*, ACM, 73–85.

[32] S. Rudich. Private communication.

[33] A. Shamir. How to Share a Secret, *Commun. Assoc. Comput. Machinery* 22 (1979), 612–613.

[34] A. Shamir. IP = PSPACE, *Proc. of FOCS 1990*, IEEE.

[35] A. C. Yao. Protocols for Secure Computations, *Proc. of FOCS 1982*, IEEE, 160–164.

[36] A. C. Yao. How to Generate and Exchange Secrets, *Proc. of FOCS 1986*, IEEE, 162–167.