Subquadratic Zero-Knowledge

JOAN BOYAR

Odense University, Odense, Denmark

GILLES BRASSARD

Université de Montréal, Montréal, Canada

AND

RENÉ PERALTA

University of Wisconsin at Milwaukee, Milwaukee, Wisconsin

Abstract. We improve on the communication complexity of zero-knowledge proof systems. Let $\mathscr{C}$ be a Boolean circuit of size n. Previous zero-knowledge proof systems for the satisfiability of $\mathscr{C}$ require the use of $\Omega(kn)$ bit commitments in order to achieve a probability of undetected cheating below 2^{-k}. In the case $k = n$, the communication complexity of these protocols is therefore $\Omega(n^2)$ bit commitments.

In this paper, we present a zero-knowledge proof system for achieving the same goal with only $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ bit commitments, where ε_n goes to zero as n goes to infinity. In the case

---

A preliminary version of this paper appeared in *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computing Science* (San Juan, P.R., Oct. 1–4). IEEE Computer Society Press, Los Alamitos, Calif., 1991, pp. 67–78.

The work of J. Boyar was supported by NSA Grant No. MDA90-H-4016. Much of this work was done while the author was employed at Loyola University of Chicago. Some of this work was done while the author was visiting Aarhus University, supported in part by the ESPIRIT II BRA Program of the EC under contract No. 3075 (Project ALCOM).

The work of G. Brassard was supported in part by NSERC's E.W.R. Stencie Memorial Fellowship and by Québec's FCAR.

The work of R. Peralta was supported in part by National Science Foundation (NSF) Grants CCR 89-09657 and CCR 92-07204.

Authors' addresses: J. Boyar, Department of Mathematics and Computer Science, Odense University, Campusvej 55, 5230 Odense M, Denmark, e-mail: joan@imada.ou.dk; G. Brassard, Département IRO, Université de Montréal, C.P. 6128, succursale centre-ville, Montréal, Québec, Canada H3C 3J7, e-mail: brassard@iro.umontreal.ca; R. Peralta, Department of Electrical Engineering and Computer Science, University of Wisconsin at Milwaukee, Milwaukee, WI 53201, e-mail: peralta@cs.uwm.edu.

$k = n$, this is $O(n\sqrt{n}^{1+\varepsilon_n})$. Moreover, only $O(k)$ commitments need ever be opened, which is interesting if it is substantially less expensive to commit to a bit than to open a commitment.

Computing Review Categories: C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*cryptographic controls*; E.3 [**Data**]: Data Encryption; F.0 [**Theory of Computation**]: General; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*alternation and nondeterminism; interactive computation; probabilistic computation*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes, F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*complexity of proof procedures; computations on discrete structures*; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs Among Complexity Measures; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*proof theory*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*probabilistic algorithms*; K.4.1 [**Computers and Society**]: Public Policy Issues—*privacy*

General Terms: Algorithms, Security, Theory

Additional Key Words and Phrases: Bit commitment, circuit evaluation, communication complexity, cryptography, interactive proofs, matrix multiplication, non-averaging sets, probabilistic algorithms, proof systems, randomness, zero knowledge

1. *Introduction*

Since the discovery that all languages in NP have polynomial-time zero-knowledge proof systems [Goldreich et al. 1991] (assuming the existence of one-way functions), the question of whether or not these protocols are efficient enough to be used in practice has become very important for cryptographers. Many practical aspects of these protocols deserve thorough investigation, such as the number of rounds, the actual computing time, the required amount of randomness, and so on. Building on the work in Boyar et al. [1993] we introduce a novel technique for decreasing the *communication complexity* of zero-knowledge proof systems for the satisfiability of Boolean circuits. In addition, we prove that this technique yields proof systems [Goldwasser et al. 1989] that are proofs of knowledge [Feige et al. 1988].

Let $\mathscr{C}$ be a satisfiable Boolean circuit of size n made out of negation and binary gates. Assume that the Prover knows a satisfying assignment for $\mathscr{C}$ and that she wishes to convince the Verifier of this fact using a zero-knowledge proof system [Goldwasser et al. 1989]. Previously known proof systems for this problem[1] all require the Prover to commit to $\Omega(n)$ bits, wait for a challenge from the Verifier, and then open some of the commitments (perhaps all of them). This is done in such a way that the Prover would be caught with probability at least 50% if in fact $\mathscr{C}$ was not satisfiable, but the Verifier learns nothing about the satisfying assignment provided that the bit commitment scheme is concealing. This process is repeated k times if the Verifier wishes a cheating prover to escape detection with probability at most 2^{-k}. (Some protocols allow the k "rounds" to take place in parallel rather than one after another. This consideration is irrelevant for the present discussion.) Therefore, *confidence level* 2^{-k} is achieved at the cost of at least $\Omega(kn)$ bit commitments. In particular, this bound becomes $\Omega(n^2)$ if we insist, as many researchers do, that the confidence level should be 2^{-n}. Other well-known zero-knowledge proof systems, such as those for Hamiltonian circuit [Blum 1987] and graph

[1]See, for example, Chaum [1987], Brassard and Crépeau [1986; 1987], and Brassard et al. [1988].

3-colorability [Goldreich et al. 1991], are not more efficient than circuit satisfiability in terms of the number of bit commitments required.

In this paper, we give a zero-knowledge proof system in which the Prover commits to $O(n^{1+\varepsilon_n})$ bits in a *preprocessing stage*, where ε_n goes to zero as n goes to infinity. Afterwards, it is enough to commit to $O(\sqrt{n}^{1+\varepsilon_n})$ additional bits in order to carry out one round of the protocol, by which a cheating prover would be caught with constant (nonzero) probability. Therefore, confidence level 2^{-k} can be achieved at the cost of $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ bit commitments, which is $O(n\sqrt{n}^{1+\varepsilon_n})$ when $k = n$, a significant improvement over the previous bound of $\Omega(n^2)$. Moreover, we show that the total number of commitments that ever have to be opened is merely $O(k)$, which is interesting if committing to a bit is significantly less expensive than opening a commitment.

Our approach could be interesting even if the confidence parameter k is taken to be a constant, despite the fact that the protocol of Brassard et al. [1988] would use only $O(n)$ bit commitments whereas our technique would use $O(n^{1+\varepsilon_n})$ of them. The reason is that the $O(n^{1+\varepsilon_n})$ commitments are necessary only in the preprocessing stage, which could take place off-line on a cheap channel that cannot support interaction efficiently. The preprocessed $O(n^{1+\varepsilon_n})$ commitments could also be published or broadcast to any number of would-be verifiers. Afterwards, only $O(\sqrt{n}^{1+\varepsilon_n})$ additional commitments are sufficient on the channel capable of efficient interaction in order to reach the desired (constant) confidence level. This could be interesting if the channel capable of interaction is much more expensive to use, or if the Prover has to give the same proof many different times to many different verifiers. A typical scenario consists of a banking card that has to prove its identity many times over to several automatic teller machines in zero-knowledge.

Our results should be compared to those concerning noninteractive zero-knowledge proof systems [De Santis et al. 1988; Blum et al. 1988; Feige et al. 1990; and others]. Those methods yield polynomial-length proofs that, assuming the existence of a trusted source of randomness, can be verified without interaction. These polynomial-length proofs can be viewed as corresponding to our preprocessing stage, with no need for the later interactive rounds. Unfortunately, even with recent improvements, greatly decreasing the length of these noninteractive proofs [Damgård 1993; Boyar and Peralta 1994; Kilian 1994], they still contain at least $O(kn \log n)$ bit commitments.

Our results should also be compared with the more recent notion of *holographic proofs* (also called *probabilistically checkable proofs*). Those techniques make it possible to rewrite a proof in a way that it can be checked very efficiently, by looking at only a polylogarithmic [Babai et al. 1991] or even constant [Arora et al. 1992] number of locations in the proof. Kilian [1992] has pointed out that holographic proofs, even though not zero-knowledge by design, can be used in a zero-knowledge context, yielding super-efficient communication complexity for each round. Using the techniques of Babai et al. [1991], Kilian showed how to produce a preprocessed proof consisting of $O(n^{1+\varepsilon})$ commitments, for arbitrarily small ε, that can be verified interactively through the use of only a polylogarithmic number $\log^{O(1/\varepsilon)}(n)$ of additional commitments per round. When n is sufficiently large, this is clearly better than the $O(\sqrt{n}^{1+\varepsilon_n})$ commitments that we require per round. If the newer holographic proofs of Arora et al. [1992] are used, Kilian's technique yields zero-knowledge proofs that require merely a constant number of bit commit-

ments per round. Under the assumption that claw-free pairs of permutations exist, Kilian makes computationally convincing *arguments* significantly more efficient, essentially eliminating the necessity of sending a preprocessed proof. However, despite very nice theoretical properties, much work still remains to be done before holographic proofs together with Kilian's technique become truly practical. This is similar to the difficulty that we had described with the use of noninteractive zero-knowledge proofs. Furthermore, contrary to our approach, it is still open whether or not Kilian's technique yields proofs of knowledge, though he conjectures that it does.

Section 2 gives an overview of bit commitment schemes and of the special property that we need in order to apply our techniques. Section 3 introduces a novel technique for verifying in zero-knowledge the validity of a matrix multiplication. This technique, which will be crucial for our application, is also interesting in its own right. Section 4 is the heart of our paper: it gives our main new technique, by which a large number of AND gates can be handled collectively in zero-knowledge at sublinear communication cost per round. More precisely, we give a *practical* scheme to handle n AND gates at the cost of $O(n^{3/5})$ bit commitments per round. Section 5 improves on the previous section from a theoretical asymptotic point of view, thanks to a technique of Szemerédi, yielding the result claimed earlier in this introduction. Section 6 shows that very few bit commitments ever have to be opened and Section 7 introduces several new bit commitment schemes that make good use of this fact. Section 8 considers statistical zero-knowledge computationally convincing protocols (arguments). Section 9 mentions some related results recently obtained by some of us and other researchers. Finally, we prove in the appendices that our technique yields not only zero-knowledge proof systems as defined by Goldwasser et al. [1989] but also proofs of knowledge as formalized by Feige et al. [1988].

2. *Required Properties of the Bit Commitment Scheme*

All known zero-knowledge protocols for NP-complete problems hinge upon the notion of a *bit commitment scheme*, also called a *blob encryption scheme*. Such a scheme allows the Prover to encrypt a bit b as a string t so that the Verifier cannot tell whether t is the encryption of a 0 or a 1; hence, the scheme is *concealing*. This commits the Prover to the bit b in the sense that later she can show the Verifier that t is the encryption of b, but she cannot falsely show that t is the encryption of the complement of b; hence the scheme is *binding*. Several implementations of such schemes are described in Boyar et al. [1990] and Brassard et al. [1988].

Formally, a *blob encryption scheme* is a function $\mathscr{E}: \Sigma \times D \to \Sigma^s$ for suitably large s and set D, where Σ denotes $\{0, 1\}$. Given $y \in \Sigma^s$, a blob *decryption* is a string $x \in D$ such that either $\mathscr{E}(0, x) = y$ (y is a 0-blob) or $\mathscr{E}(1, x) = y$ (y is a 1-blob). Whenever the Prover gives the Verifier a decryption for a blob, she is said to *open* that blob. Given bit b, we shall denote by $F(b)$ a blob used to commit to b, that is $F(b) = \mathscr{E}(b, x)$ for an x randomly chosen in D. We extend this notation in the obvious way to $F(\vec{x})$ and $F(M)$, where $\vec{x}$ and M are bit vectors and matrices, respectively.

Blob encryption schemes must satisfy several standard properties [Boyar et al. 1990; 1993; Brassard et al. 1988]. In addition, for our techniques to apply,

it is crucial that the blobs used allow *noninteractive processing* of XOR gates. A blob scheme allows noninteractive processing of a particular type of gate if the Verifier, given blobs for the input to a gate of that type, can create a valid blob for the output of that gate, without requiring any help from the Prover. Moreover, the Prover must also be able to compute the very same blob without interaction with the Verifier. The output blob must be such that opening it does not release any information on the input blobs (other than whatever can be learned from knowing the Boolean output of the gate). Notice that blobs that offer noninteractive processing of XOR gates also automatically offer noninteractive processing of NOT gates. For this, the Prover offers and immediately opens a *reference 1-blob* for the Verifier. The negation of any further blob can then be noninteractively computed by XORing it with the reference blob.

The following bit commitment scheme, which is based on the "Quadratic Residuosity Assumption" [Goldwasser and Micali 1984] has all the properties we require.

Definition 2.1. *A Blum integer* is the product of two primes P and Q of equal bit-length such that $P \equiv Q \equiv 3 \pmod 4$.

Definition 2.2 (*Brassard-Crépeau* [1987]). Let N be a fixed Blum integer produced by the Prover. We define *QR-blobs* by $\mathscr{E}(0, x) = x^2 \bmod N$ and $\mathscr{E}(1, x) = -x^2 \bmod N$, for $x \in \mathbb{Z}_N^*$. (Before this bit commitment scheme can be used, the Prover must convince the Verifier that N is a Blum integer [van de Graaf and Peralta 1988]. In this paper, we neglect the communication complexity of the subprotocol used for this preliminary task.)

Note that 0-blobs are quadratic residues and 1-blobs are quadratic nonresidues, because -1 is always a quadratic nonresidue modulo a Blum integer. Hence, QR-blobs are unconditionally binding for the Prover: It is not possible for her to change a commitment (unless she was sufficiently daring and lucky to get away with an N that is not a Blum integer). On the other hand, the concealing security of QR-blobs depends on the Quadratic Residuosity Assumption, which states that having any significant advantage in deciding quadratic residuosity is infeasible for random elements of $\mathbb{Z}_N^*$ whose Jacobi symbol is $+1$, whenever N is a sufficiently large Blum integer of unknown factorization. (QR-blobs always have Jacobi symbol $+1$ because such is the case of -1 modulo a Blum integer.)

It is easy to see that QR-blobs allow noninteractive processing of XOR gates. If one multiplies two or more blobs together, the Prover, who created these blobs, can decrypt the result as bit b if and only if the XOR of the bits represented by those blobs is that bit b. Thus, given the blobs representing the inputs to a XOR gate, the Prover and the Verifier can each compute a blob representing the output. They can be certain that the other computed the same output blob, and no interaction is required to do this.

In the following three sections, as well as in Appendices B and C, we shall assume that QR-blobs are used, although other bit commitment schemes can also be used (for instance, see Sections 7 and 8).

3. *Proving Matrix Multiplication*

Consider three $m \times m$ matrices A, B, and C over **GF**(2). Let $F(A)$, $F(B)$, and $F(C)$ be blob-based encryptions of these matrices, which are used by the

Prover to commit to the contents of these matrices. Assume that the Prover would like to convince the Verifier in zero-knowledge that C is the product of A and B. Known circuits for matrix multiplication have size complexity $\Omega(m^c)$ with c about 2.376 [Coppersmith and Winograd 1990]. However, for m in the practical range, one cannot do much better than $\Omega(m^3)$. Thus the communication cost of a straightforward zero-knowledge proof that $C = AB$ is $\Omega(m^3)$ blobs per round. We show how to achieve the same result while communicating only $O(m)$ blobs per round.

Although it is unknown if two matrices can be multiplied in time $O(m^2)$, it is known that matrix multiplication can be *checked* with arbitrarily small constant error probability in time $O(m^2)$ if one allows coin-flipping. This probabilistic algorithm, due to Freivalds [1979], is rather simple. A vector $\vec{y}$, consisting of m elements from **GF**(2), is chosen at random from the uniform distribution. Then, the two m-bit vectors, $C \cdot \vec{y}$ and $A \cdot (B \cdot \vec{y})$, are compared. If $C = AB$, these vectors will always be identical. If $C \neq AB$, there is at least a 50% chance that these vectors will differ.

Let us consider choosing a second m-bit vector $\vec{x}$, also at random from the uniform distribution, and checking that $\vec{x} \cdot C \cdot \vec{y} = \vec{x} \cdot A \cdot B \cdot \vec{y}$. It is easy to see that this also leads to a probabilistic algorithm with running time $O(m^2)$ for checking matrix multiplication, although this would be uninteresting in the conventional algorithmic setting.

LEMMA 3.1. *Suppose A, B, and C are $m \times m$ matrices over the field* **GF**(2) *and suppose that $\vec{x}$ and $\vec{y}$ are m-bit vectors over* **GF**(2), *chosen randomly from the uniform distribution. If $C = AB$, then $\vec{x} \cdot C \cdot \vec{y} = \vec{x} \cdot A \cdot B \cdot \vec{y}$, and if $C \neq AB$, then the probability that $\vec{x} \cdot C \cdot \vec{y} \neq \vec{x} \cdot A \cdot B \cdot \vec{y}$ is at least* $1/4$.

PROOF. It is clear that if $C = AB$, then $\vec{x} \cdot C \cdot \vec{y} = \vec{x} \cdot A \cdot B \cdot \vec{y}$, so assume that $C \neq AB$. Let $\vec{u} = C \cdot \vec{y}$ and $\vec{v} = A \cdot B \cdot \vec{y}$. From Freivalds [1979], we know that there is a probability of at least $1/2$ that $\vec{u} \neq \vec{v}$. If this is the case, techniques similar to those of Freivalds [1979] show that exactly half of the possible $\vec{x}$ vectors will result in $\vec{x} \cdot \vec{u} \neq \vec{x} \cdot \vec{v}$. This implies that with probability at least $1/4$, $\vec{x} \cdot C \cdot \vec{y} \neq \vec{x} \cdot A \cdot B \cdot \vec{y}$. □

At first glance, this does not appear to help us since it requires an additional dot product to reach an inferior confidence level. But remember that we use blobs that allow for noninteractive processing of XOR gates. Thus, the Verifier can choose the vectors $\vec{x}$ and $\vec{y}$, and he can compute $F(\vec{x} \cdot (C \cdot \vec{y}))$, $F(\vec{x} \cdot A)$ and $F(B \cdot \vec{y})$ noninteractively from $F(A)$, $F(B)$ and $F(C)$. To compute these products, he does several dot products of a known vector ($\vec{x}$ or $\vec{y}$) with a blobbed vector. Each dot product is easy because ones in the known vector determine which elements of the blobbed vector should be in the XOR, which is computed simply by multiplying together the selected blobs (if QR-blobs are used). This means that the Verifier needs help from the Prover only to compute $F(\vec{x} \cdot A \cdot B \cdot \vec{y})$ from $F(\vec{x} \cdot A)$ and $F(B \cdot \vec{y})$, and to compare that with $F(\vec{x} \cdot C \cdot \vec{y})$.

But to compute $\vec{x} \cdot A \cdot B \cdot \vec{y}$ from $\vec{x} \cdot A$ and $B \cdot \vec{y}$, one simply does a dot product, which is a XOR of ANDs, since those are the operations in **GF**(2). If the Verifier sends the Prover $\vec{x}$ and $\vec{y}$, a linear number of bits, the Prover can then supply blobs $F(z_i)$ for $z_i = (\vec{x} \cdot A)_i \wedge (B \cdot \vec{y})_i$, $1 \leq i \leq m$. Using a single round of the techniques of Brassard and Crépeau [1987] and Boyar et al.

[1993], she can convince the Verifier of the validity of these ANDs with probability at most $1/2$ of successful cheating. Once the Verifier has the $F(z_i)$'s, he can noninteractively compute $Z = F((\vec{x} \cdot C \cdot \vec{y}) \oplus (\bigoplus_{i=1}^{m} z_i))$. At this point, the honest Prover can open Z to prove that it encodes 0, thus showing that $\vec{x} \cdot A \cdot B \cdot \vec{y} = \vec{x} \cdot C \cdot \vec{y}$. If, in fact, $C \neq AB$ and $\vec{x} \cdot A \cdot B \cdot \vec{y} \neq \vec{x} \cdot C \cdot \vec{y}$, then a cheating prover maximizes her probability of not getting caught by cheating on a single AND. Thus, the overall probability that the Prover will be caught cheating by the Verifier in any given round is at least $1/8$ since the $1/4$ from Lemma 3.1 is multiplied by the probability $1/2$ of catching the Prover cheating on that AND.

A detailed description of the protocol is given in Appendix B, as well as a proof that it is zero-knowledge. Using this protocol, the communication complexity for checking a matrix product of $m \times m$ matrices involves $O(m)$ blobs per round, and $O(k)$ rounds suffice to achieve confidence level 2^{-k}.

4. *Proving Any Circuit in ZK with Sublinear Bit Commitments per Round*

Any Boolean circuit with arbitrary binary gates and NOT gates can easily and efficiently be converted into a circuit that contains only binary AND gates and NOT gates,[2] in such a way that the size of the new circuit is linearly related to the size of the original circuit and the output of a NOT gate is never the input to another NOT gate. Let n be the number of AND gates in the new circuit. Figure 1 illustrates that the circuit can be thought of as a set of n equations of the form $c_i = a_i \wedge b_i$, where a_i and b_i correspond to the Boolean inputs to the ith AND gate and c_i is its output. In the example, we use x variables to denote Boolean values carried by the circuit wires (excluding the output of NOT gates, which are denoted by $\bar{x}$ variables).

The Prover, who claims to know a satisfying assignment for the circuit, gives one blob to the Verifier for each input wire and for the output wire of each AND gate in the circuit. If the Prover is honest, she does this in a way that is consistent with her known satisfying assignment. The output blobs of NOT gates are not given explicitly; rather, both the Prover and the Verifier compute them noninteractively from the input blobs of the same gates. The Prover also opens the blob on the final output wire of the circuit, in order to show that it encrypts a 1. For each i, let $F(a_i)$ and $F(b_i)$ be the blobs given as inputs to the ith AND gate, and let $F(c_i)$ be the output blob for that gate. Now, the Prover's task reduces to convincing the Verifier that $c_i = a_i \wedge b_i$ given blobs $F(a_i)$, $F(b_i)$ and $F(c_i)$, for $1 \leq i \leq n$. In this section, we show that these n AND gates can be handled collectively in sublinear communication complexity per round, at the cost of superlinear preprocessing.

For this, we use matrix multiplication. The a_i's will be in a matrix A, the b_i's in a matrix B, and the c_i's in a matrix C. The challenge is to place the a_i's and b_i's in A and B such that the product matrix AB contains the c_i's in isolation. Obviously, if the a_i's and the b_i's are on the diagonals of their respective matrices, and if the other entries of these matrices are 0, the c_i's will be on the diagonal of the product. However, this would not yield any improvement over the straightforward protocol. Instead, we need to put the a_i's and b_i's in

[2] However, it would be preferable in practice to keep XOR gates rather than transform them into AND and NOT gates since XOR gates can be processed noninteractively.

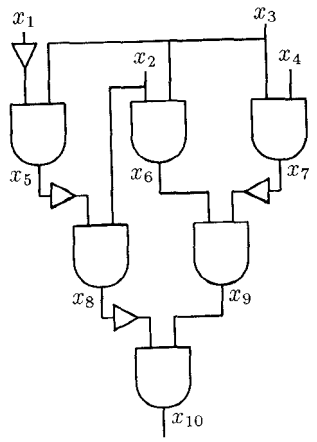

FIG. 1. Transforming a circuit into equations.

$$x_5 = \bar{x}_1 \wedge x_3 \qquad x_8 = x_2 \wedge \bar{x}_5$$
$$x_6 = x_2 \wedge x_3 \qquad x_9 = x_6 \wedge \bar{x}_7$$
$$x_7 = x_3 \wedge x_4 \qquad x_{10} = \bar{x}_8 \wedge x_9$$

$m \times m$ matrices, where m is significantly smaller than n. (Clearly, we cannot reduce m below $\sqrt{n}$.) Using the techniques of Section 3, this reduces the communication complexity for one round of checking the n AND gates to $O(m)$ blobs. Unfortunately, the product matrix tends to have many "garbage" elements, which the Verifier cannot compute on his own from the blobs for the a_i's, b_i's and c_i's.

For instance, consider the following matrices, with $n = 6$ and $m = 3$:

$$\begin{pmatrix} a_1 & a_2 & 0 \\ a_3 & 0 & a_4 \\ 0 & a_5 & a_6 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} b_3 & b_1 & 0 \\ b_5 & 0 & b_2 \\ 0 & b_6 & b_4 \end{pmatrix}.$$

Multiplying the two matrices over **GF**(2) (i.e., + and juxtaposition stand for XOR and AND, respectively), one obtains

$$\begin{pmatrix} a_1b_3 + a_2b_5 & a_1b_1 & a_2b_2 \\ a_3b_3 & a_3b_1 + a_4b_6 & a_4b_4 \\ a_5b_5 & a_6b_6 & a_5b_2 + a_6b_4 \end{pmatrix}.$$

The product matrix contains all the c_i's, but it also contains three other elements $\gamma_1 = a_1b_3 + a_2b_5$, $\gamma_2 = a_3b_1 + a_4b_6$, and $\gamma_3 = a_5b_2 + a_6b_4$. In this case, given the blobs for the a_i's, b_i's, and c_i's, both the Prover and the Verifier can create $F(A)$ and $F(B)$. (A standard blob—such as 1 for QR-blobs—is used for the explicit zeros in the matrices, so that both the Prover and the Verifier end up with exactly the same $F(A)$ and $F(B)$ matrices.) However, the Prover has to supply blobs for the "garbage elements" γ_1, γ_2, and γ_3 for the Verifier to obtain $F(C)$. The key point is that these additional blobs, along with the blobs for the commitments to the values on all the wires in the circuit,

can be given to the Verifier in the *preprocessing stage*, which is carried out once and for all, independently of the desired confidence level (i.e., of the number of rounds).

Continuing the example of Figure 1, here is the matrix product corresponding to our circuit, where the asterisks are used to denote the garbage terms.

$$\begin{pmatrix} \bar{x}_1 & x_2 & 0 \\ x_3 & 0 & x_2 \\ 0 & x_6 & \bar{x}_8 \end{pmatrix} \times \begin{pmatrix} x_4 & x_3 & 0 \\ \bar{x}_7 & 0 & x_3 \\ 0 & x_9 & \bar{x}_5 \end{pmatrix} = \begin{pmatrix} * & x_5 & x_6 \\ x_7 & * & x_8 \\ x_9 & x_{10} & * \end{pmatrix}.$$

Any set of Boolean values for the free variables (including the asterisks) that makes the matrix product come out correctly provides a satisfying assignment for the circuit, as long as the output wire x_{10} is assigned value 1. For instance,

$$x_{1..10} = 0,1,1,0,1,1,0,0,1,1$$

shows that the circuit is satisfiable since the matrix product

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} * & 1 & 1 \\ 0 & * & 0 \\ 1 & 1 & * \end{pmatrix}$$

is correct when the garbage terms are set accordingly.

It is shown in Appendix A that this $n = 6$ and $m = 3$ template can be improved to $n = 32$ and $m = 8$. This can be used recursively, by viewing each entry as a submatrix. Using this tensor product construction, one can put 32^t entries in an $8^t \times 8^t$ matrix, for any positive integer t. Thus, it is possible to put $n = m^{5/3}$ entries in an $m \times m$ matrix if m is a power of 8. This gives a communication complexity of $O(n^{3/5})$ blobs per round for handling n AND gates. The price to pay for this increased efficiency in each round is that the number of garbage elements in these matrices is $O(n^{6/5})$, which sets the communication cost of the preprocessing stage. Therefore, any Boolean circuit with n arbitrary binary gates and NOT gates can be verified at confidence level 2^{-k} with communication complexity $O(n^{6/5} + kn^{3/5})$ blobs. It is proven in the appendices, under the Quadratic Residuosity Assumption, that this protocol is a zero-knowledge proof of knowledge.

The above result is our best from a practical point of view. However, an asymptotic improvement of theoretical interest has been discovered by Endre Szemerédi. The next section shows how it is possible to put n entries in matrices of size $m \times m$, where $m \leq \sqrt{n/2}^{1+\varepsilon_n}$ and $\varepsilon_n = 4\sqrt{2} / \sqrt{\lg(n/2)}$, provided that $n \geq 2^{129}$. This yields the result claimed in the introduction since ε_n goes to zero, albeit slowly, as n goes to infinity. Although the above formula seems to indicate that Szemerédi's improvement does not overtake recursive use of our 8×8 construction until n is at least roughly 2^{800} (corresponding to $\varepsilon_n = 1/5$), a more careful analysis shows that it allows one to put 2^{135} entries in $2^{80} \times 2^{80}$ matrices, whereas the 8×8 matrix used recursively would require matrices of size $2^{81} \times 2^{81}$ to handle as many entries. Although the exact crossover point is still an open question, we doubt that it is in the practical range.

5. *Szemerédi's Improvement*

The key to Szemerédi's improvement lies in the seemingly irrelevant notion of a *nonaveraging set*.

Definition 5.1. A set S of integers is *nonaveraging* if, for any distinct x and y in S, it is not the case that their arithmetic average is also in S:

$$(\forall x, y \in S)\left[x \neq y \Rightarrow \frac{x+y}{2} \notin S\right].$$

Equivalently, the set S does not contain three distinct integers that are in arithmetic progression.

For instance, $\{0, 1, 3, 4, 9\}$ is nonaveraging, but it would not be if any additional single-digit integer were added. Let $\mathbb{N}$ stand for the set of natural numbers. Let $\nu: \mathbb{N} \to \mathbb{N}$ be such that $\nu(m)$ denotes the cardinality of the largest possible nonaveraging set of integers between 0 and $m-1$. For instance, we have just seen that $\nu(10) \geq 5$ (and in fact $\nu(10) = 5$).

Consider any integer m. Let $t \leq \nu(m)$, $n = 2mt$, and let $S = \{s_0, s_1, \ldots, s_{t-1}\}$ be a nonaveraging set of integers between 0 and $m-1$. Let $a_0, a_1, \ldots, a_{n-1}$, $b_0, b_1, \ldots, b_{n-1}$ and $c_0, c_1, \ldots, c_{n-1}$ be as in Section 4. (It is easier to start the indices at 0.) In this section, let "$\oplus$" and "$\ominus$" stand for addition and subtraction modulo $2m$, respectively. We build $2m \times 2m$ matrices A and B (whose indices range from 0 to $2m-1$), as follows: Consider any integer k between 0 and $n-1$. Decompose k as $i + 2mj$, where $0 \leq i \leq 2m-1$ and $0 \leq j \leq t-1$. Simply put a_k in $A[i \oplus s_j, i]$ and b_k in $B[i, i \oplus 2s_j]$. The other entries of A and B are set to 0.

To show that A is well defined, we must argue that a_k and a_l are placed in different positions if $k \neq l$. Let $k = i_k + 2mj_k$ and $l = i_l + 2mj_l$. If $i_k = i_l$ (otherwise, the claim is obvious), then, since $k \neq l$, we have $j_k \neq j_l$ and therefore $s_{j_k} \neq s_{j_l}$. Since $0 \leq s_{j_k} < m$ and $0 \leq s_{j_l} < m$, we have $s_{j_k} \not\equiv s_{j_l} \pmod{2m}$ and therefore $i_k \oplus s_{j_k} \neq i_l \oplus s_{j_l}$. Thus $(i_k \oplus s_{j_k}, i_k) \neq (i_l \oplus s_{j_l}, i_l)$. To see that B is well defined, note that if $i_k = i_l$, then again $s_{j_k} \neq s_{j_l}$. Since $0 \leq 2s_{j_k} < 2m$ and $0 \leq 2s_{j_l} < 2m$, we have $2s_{j_k} \not\equiv 2s_{j_l} \pmod{2m}$ and therefore $i_k \oplus 2s_{j_k} \neq i_l \oplus 2s_{j_l}$. Thus $(i_k, i_k \oplus 2s_{j_k}) \neq (i_l, i_l \oplus 2s_{j_l})$.

We now show that if $C = AB$, then $C[i \oplus s_j, i \oplus 2s_j]$ is $c_k = a_k b_k$, where $k = i + 2mj$. It is clear that $a_k b_k$ appears as a term in $C[i \oplus s_j, i \oplus 2s_j]$. It remains to be verified that $a_k b_k$ is not smeared by garbage. But for garbage to occur, it would have to be the case that there exists an integer u, $0 \leq u \leq 2m-1$, such that $A[i \oplus s_j, u]$ and $B[u, i \oplus 2s_j]$ are simultaneously nonzero, with $u \neq i$. This would imply the existence of integers v and w different from j, such that $i \oplus s_j = u \oplus s_v$ and $i \oplus 2s_j = u \oplus 2s_w$. Subtracting the first equation from the second yields $s_j = 2s_w \ominus s_v$, and hence $2s_w = s_j \oplus s_v$. But then $2s_w = s_j + s_v$ since s_j, s_v and s_w are smaller than m and the addition is performed modulo $2m$. We conclude that s_w is the arithmetical average of s_j and s_v, for $j \neq v$, which is a contradiction.

It is therefore possible to use the technique of Section 4 to handle collectively up to $n = 2m\nu(m)$ ANDs by proving the product of $2m \times 2m$ matrices. In order to determine if this is better than $n = (2m)^{5/3}$, which is obtained through recursive use of the 8×8 matrix product given in Appendix A (provided that $2m$ is a power of 8), a lower bound on $\nu(m)$ is necessary.

Fortunately, this is a classic problem: It has been known since 1942 that $\nu(m)$ is $\Omega(m^{1-o(m)})$, a (nonconstructive) result due to Salem and Spencer [1942]. Therefore, $2m\nu(m)$ is $\Omega(m^{2-o(m)})$, which is clearly better than $(2m)^{5/3}$ for sufficiently large m.

From a "practical" point of view, it is interesting to find explicit (as opposed to asymptotic and nonconstructive) bounds on $\nu(m)$ in order to determine the value of n at which Szemerédi's technique becomes preferable. For this, we start from the work of Moser [1953] who was the first to give explicit bounds on $\nu(m)$. Refining his technique, we show below that $\nu(2^{79}) \geq 2^{55}$. This establishes that we can handle 2^{135} ANDs using $2^{80} \times 2^{80}$ matrices, whereas recursive use of the 8×8 matrix would require matrices of size $2^{81} \times 2^{81}$ to handle as many ANDs. It is not known if Szemerédi's approach could yield an improvement in the practical range through better explicit bounds on $\nu(m)$.

LEMMA 5.2. *For any integers* $k \geq 1$, $l \geq 1$, *and* $i \geq 0$,

$$\nu(3^i 2^{k(l+1)+2l+\lceil \lg k \rceil - 1}) \geq 2^{kl+i}.$$

In particular $\nu(2^{79}) \geq 2^{55}$, *using* $k = 11$, $l = 5$, *and* $i = 0$.

PROOF. The fact that $\nu(2^{79}) \geq 2^{55}$ is shown as follows: Consider any $x \in \{0,1\}^{55}$. Decompose x into 11 blocks of 5 bits: $x = x_1 x_2 \cdots x_{11}$, where each $x_i \in \{0,1\}^5$ is considered as an integer between 0 and 31. Let y be the integer whose binary representation is $z0x_1 0x_2 0 \cdots 0x_{11}$, where the 0's are single-bit zeros and z is the integer computed as

$$z = \sum_{i=1}^{11} \frac{x_i(x_i + 1)}{2}.$$

Note that $z \leq 5456$; hence, 13 bits suffice to represent it. Therefore, the bit-length of y is at most $13 + 11 \times (1 + 5) = 79$. Let S be the set of y's thus formed. We claim S is nonaveraging. Let $a, b \in S$, where $a = z_a 0 a_1 0 a_2 0 \cdots 0 a_{11}$ and $b = z_b 0 b_1 0 b_2 0 \cdots 0 b_{11}$. If the average $c = (a + b)/2$ is of the form $c = z_c 0 c_1 0 c_2 0 \cdots 0 c_{11}$, then $c_i = (a_i + b_i)/2$ and

$$z_c = \frac{z_a + z_b}{2} = \sum_{i=1}^{11} \frac{a_i^2 + b_i^2 + a_i + b_i}{4}.$$

It is easy to check that this summation majorizes $\sum_{i=1}^{11} c_i(c_i + 1)/2$, equality holding if and only if $a_i = b_i$ for all i, and thus $z_a = z_b$ as well. Therefore, $c \in S$ is possible only if $a = b$, as claimed.

Clearly, S contains 2^{55} integers smaller than 2^{79}, which establishes $\nu(2^{79}) \geq 2^{55}$. [In fact, the largest element in S is about $\frac{2}{3} \times 2^{79}$; the precise result is $\nu(402618050541861698664416) \geq 2^{55}$]. To obtain the more general result, use k groups of l bits (instead of 11 groups of 5 bits) and use i times the easily-proven fact that $\nu(3n) \geq 2\nu(n)$. We leave the details to the reader. □

As long as $k \geq 8$, using the facts that $k \geq 5 + \lceil \lg k \rceil$ and $2^5 \geq 3^3$, Lemma 5.2 with $i = 3$ yields $\nu(2^{kl+2(k+l)-1}) \geq 2^{kl+3}$. Thus $2^{2(k+1)(l+1)+1}$ ANDs can be

handled collectively using matrices of size $2^{kl+2(k+l)}$. Let n be the number of ANDs to be handled. Taking

$$k = l = \left\lfloor \sqrt{\lg \sqrt{n/2}} \right\rfloor$$

yields

$$2^{2(k+1)(l+1)+1} > n \qquad \text{and} \qquad 2^{kl+2(k+l)} \leq \sqrt{n/2}^{1+\varepsilon_n},$$

where $\varepsilon_n = 4\sqrt{2} / \sqrt{\lg(n/2)}$, which proves the claim at the end of Section 4. Note that these bounds hold for $n \geq 2^{129}$, since this is implied by

$$k = l = \left\lfloor \sqrt{\lg \sqrt{n/2}} \right\rfloor \geq 8.$$

6. *How to Open Only a Very Few Commitments*

Until now, we have measured communication complexity in terms of the number of *bit commitments* that are exchanged between the Prover and the Verifier. We shall now concentrate on the actual number of *bits* exchanged during the interaction. For this purpose, we shall apply our technique more carefully, and distinguish between the number of bit commitments and the number of those commitments that need be opened. It turns out that the communication complexity for proving any circuit in zero-knowledge (including the preprocessing stage) can be reduced to $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ bit commitments plus $O(k)$ openings of commitments.

This would be interesting if it were substantially less expensive to commit to a bit than to open a commitment. Bit commitment schemes that cost only an amortized constant number of communication bits per commitment have been proposed previously by Naor [1990] and Kilian et al. [1989] but unfortunately those schemes cannot be used in our context because they do not provide for noninteractive processing of XOR gates. Although we are unable to prove their security based on standard cryptographic assumptions, we describe in Section 7 several heuristic bit commitment schemes that provide for noninteractive processing of XOR gates, for which each commitment costs only an amortized constant number of bits in communication complexity, whereas each opening of a commitment is no more expensive than with the classic QR-blobs. Therefore, if it takes no more than $O(\sqrt{n}^{1+\varepsilon_n})$ bits to open a commitment, the resulting communication complexity is reduced from $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ blobs to $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ bits, which is $O(n\sqrt{n}^{1+\varepsilon_n})$ bits, if we take $k = n$. The reader is invited to prove the security of our new commitment schemes (under a suitable cryptographic assumption) or find other provable schemes with the same efficiency.

The communication complexity of the preprocessing stage described in the previous section is dominated by the blobs used for the garbage elements, which is $O(n^{1+\varepsilon_n})$ bit commitments. After that, $O(k)$ rounds of proving a matrix multiplication are performed. In each such round, the Verifier chooses the $\vec{x}$ and $\vec{y}$ vectors, each of which is $O(\sqrt{n}^{1+\varepsilon_n})$ bits long, and sends them to the Prover. Then, both the Prover and the Verifier noninteractively compute

blobs for the vectors $\vec{x} \cdot A$ and $B \cdot \vec{y}$ and for the element $\vec{x} \cdot C \cdot \vec{y}$. At this point, it remains for the Prover to convince the Verifier of the validity of a dot product involving $O(\sqrt{n}^{1+\varepsilon_n})$ blobs. To do this, the Prover sends the Verifier blobs for the outputs of the $O(\sqrt{n}^{1+\varepsilon_n})$ AND gates together with auxiliary blobs for checking these outputs. Using the techniques of Boyar et al [1993], three auxiliary blobs are sufficient for each AND gate. Therefore, $O(\sqrt{n}^{1+\varepsilon_n})$ bit commitments are sufficient so far.

At this point, the Verifier issues one single-bit challenge.[3] For each of the AND gates, the Prover must respond to this challenge either by showing that two of the auxiliary blobs are encryptions of the same bits as the gate's inputs and the other is the encryption of a 0, or by showing that two of the auxiliary blobs are encryptions of the same bit as the gate's output (see Boyar et al. [1993] for details). As the first step toward answering this challenge, the Prover needs to either specify which of the auxiliary blobs correspond to which inputs or to specify which of the auxiliary blobs are encryptions of the same bit as the output. A constant number of bits per gate suffices to convey this information. It remains for the Prover to show that some blobs are equal in pairs, and perhaps that some others are encryptions of 0. But showing that two blobs encrypt the same bit is equivalent to showing that their noninteractively computed XOR encrypts 0. Thus, the Prover and Verifier can noninteractively construct a vector $F(\vec{u})$ of $O(\sqrt{n}^{1+\varepsilon_n})$ blobs, which the Prover must then show to be an encryption of the zero vector. (The last blob in $F(\vec{u})$ is the blob Z introduced at the end of Section 3.)

In order to prove that a vector of blobs contains only encryptions of zeros without opening them all, we resort once again to Freivalds' technique. More precisely, the Verifier chooses a random bit vector $\vec{z}$ of the same length as $\vec{u}$, and he sends it to the Prover as a challenge. Both the Prover and the Verifier noninteractively compute the XOR of the blobs in $F(\vec{u})$ corresponding to ones in $\vec{z}$. Finally, the Prover opens the resulting blob to show that it encrypts a 0. *This is the one and only commitment that is opened during this round.* The Prover will be caught cheating with probability exactly 50% if it is not the case that all the blobs in $F(\vec{u})$ encrypt zero. Therefore, there is at least one chance in sixteen of catching the Prover if she is cheating on this round, since the $1/8$ from Section 3 is multiplied by this 50%.

In summary, this proof system requires the Prover to commit to $O(n^{1+\varepsilon_n})$ bits during the preprocessing stage and to an additional $O(\sqrt{n}^{1+\varepsilon_n})$ bits per round, but only a single commitment need be opened in each round. In addition, $O(\sqrt{n}^{1+\varepsilon_n})$ bits per round are sent in the clear between the Prover and the Verifier, but this can be neglected in the face of the number of bit commitments. Since the total number of rounds to achieve confidence level 2^{-k} is $O(k)$, the total communication complexity of this protocol is $O(n^{1+\varepsilon_n} + k\sqrt{n}^{1+\varepsilon_n})$ bit commitments and the opening of $O(k)$ of them.

The proof that this approach yields a zero-knowledge proof of knowledge is similar to that for the simpler protocol of Section 4, which is given in the appendices.

It should be pointed out that there exists an even more efficient way for the Prover to convince the Verifier that a vector $F(\vec{u})$ of l blobs encrypts a known

[3]There is no need for the Verifier to issue an independent challenge for each AND gate, but of course he must issue one independent challenge at each round.

vector $\vec{v}$. We leave it for the reader to determine how the use of *epsilon-biased* random variables [Naor and Naor 1990; Peralta 1990] allows for constant error probability with the exchange of merely $O(\log l)$ bits plus the opening of a constant number of blobs (rather than $O(l)$ bits with the technique above). Unfortunately, this improvement would not change the global analysis of our proof system, which is why we did not use it.

7. *Cheap Blobs*

In this section, we present some heuristic techniques for lowering the communication complexity of our proof systems. It is an open question whether any of these techniques yield secure protocols. However, we chose to include them because, among many possible such heuristics, these have at least some supporting theoretical results in the computer science literature.

Let m stand for the total number of bits to which the Prover will need to commit during the protocol (it is not necessary that the Prover knows the value of m before starting the protocol). If one uses QR-blobs, the communication cost of committing to a bit is roughly the same as the cost of opening a commitment. For the approach of Section 6 to be worthwhile, we need a bit commitment scheme for which it is much cheaper to commit to a bit than to open a commitment. We now sketch variations on such a scheme, by which the Prover can commit to m bits at the cost of communicating only $O(m)$ bits to the Verifier, and such that the cost of opening one commitment is essentially the same as for QR-blobs. Moreover, these bit commitment schemes allow the Prover to commit to her bits one by one (there is no need to commit to them all at once), to open precisely those that she wishes to open, also one by one, and more importantly, they provide for noninteractive processing of XOR gates. The general idea is to give a short description that can be efficiently converted into a sequence of m QR-blobs. This can be achieved in at least three different ways, but we must stress again that none of these techniques have been proven secure. It is an interesting open question whether or not these schemes are computationally concealing assuming only the Quadratic Residuosity Assumption.

The first idea uses a cryptographically secure pseudorandom bit generator [Blum and Micali 1984; Yao 1982] to generate enough pseudorandom bits to produce the m QR-blobs. Initially, the Prover and Verifier set up a QR-blob scheme: the Prover chooses a Blum integer N whose factorization she privately knows, and she convinces the Verifier that it is a Blum integer. Let β stand for the bit length of N. In order to set up the new bit commitment scheme, the Prover selects a cryptographically secure pseudorandom bit generator and a seed of length m for this generator. She sends the seed to the Verifier. Each time the Prover wishes to commit to a bit b, she and the Verifier compute the next β bits in the pseudorandom sequence generated from the seed. If the resulting bit string, considered as an integer, is larger than $N-1$, or if its Jacobi symbol modulo N is not $+1$, both the Prover and the Verifier generate the next β bits until an element $x \in \mathbb{Z}_N^*$ is obtained whose Jacobi symbol is $+1$. (This process cannot take too long because the probability of success must be at least 25% for otherwise the pseudorandom generator would be broken.) At this point, both the Prover and the Verifier have arrived at the same x without any need for interaction. In order to commit to b, the Prover then

sends a single bit c such that b is the XOR of c and of the QR-blob decryption of x. (Note that the Prover can decrypt QR-blobs because she knows the factorization of N.) Thus, we see that $O(1)$ bits of communication is sufficient per commitment, in the amortized sense, and that β bits must be communicated to open a commitment. Moreover, the reader can easily verify that this bit commitment scheme provides for noninteractive computation of XOR gates.

It is clear that this bit commitment scheme is unconditionally binding for the Prover (provided N is a Blum integer), but is it computationally concealing? Let us first note that we do not stretch the pseudorandom sequence unduly. With probability at least $1/4$, each pseudorandom subsequence of β bits is smaller than N and has Jacobi symbol $+1$, because otherwise this could be used to distinguish the pseudorandom sequence from a truly random sequence. Therefore, we need generate only $O(m\beta)$ pseudorandom bits. But β is certainly polynomial in m, for otherwise the Verifier could not receive even a single commitment in polynomial time—even if standard QR-blobs were used —since m is polynomially related to the circuit's size. Hence, the number of pseudorandom bits that need be generated is polynomial in m, the size of the seed. Therefore, assuming that the pseudorandom generator is secure and assuming the Quadratic Residuosity Assumption, the Verifier would have no polynomial-time advantage in guessing the quadratic residuosity of the pseudorandom substrings that are kept *if the seed were kept secret*. However, it is necessary for the Prover to reveal the seed to the Verifier. Once this is done, the security of the protocol is plausible, but no longer a theorem under standard cryptographic assumptions.

Our second candidate for a bit commitment scheme with the desired properties is very similar to the first, except that we produce the required number of blocks of β bits from a short public seed through the use of epsilon-biased random variables [Naor and Naor 1990; Peralta 1990], in which case we need so little expansion that the bias ε could be almost exponentially small. Hence, we trade cryptographic unpredictability for a lower rate of expansion of the seed and for the almost-uniform distribution guarantee given by the theory of epsilon-biased random variables.

Finally, Mihir Bellare (personal communication) has suggested a nice simplification of the above template. Let the Prover and the Verifier agree again on the parameters of a QR-blob scheme. In addition, the Prover chooses a random $x \in \mathbb{Z}_N^*$ and gives it to the Verifier. When committing to a bit, rather than using a pseudorandom bit generator to produce the next number with Jacobi symbol $+1$, one can use the next such number from the sequence $x+1, x+2, \ldots$ in $\mathbb{Z}_N^*$. Notice that the Prover and the Verifier can easily produce the same numbers, but only the Prover can compute the QR-blob decryptions of them (assuming the Quadratic Residuosity Assumption). The Prover can then commit to her bit in the manner described above. The problem with proving the validity of this approach, of course, is that it is not known (even under the Quadratic Residuosity Assumption) whether or not any information about the *joint distribution* of the bits committed to can be obtained efficiently from x without knowledge of the factors of N. It is conceivable that opening some of the blobs would give information about the bits hidden in other blobs, because the elements of $\mathbb{Z}_N^*[+1]$ are not chosen independently, or that joint information about several hidden bits could be

computed efficiently by the Verifier. Nevertheless, recent results by Bach [1991], Damgård [1990], Alon et al. [1992], and Peralta [1992] on the Legendre and Jacobi symbols of consecutive numbers provide a beginning of a theoretical justification for this scheme.

8. *Statistical Zero-Knowledge Arguments*

Throughout this paper, we have used blobs that are unconditionally binding but merely computationally concealing. The techniques of Sections 3, 4, and 5, however, do not require this. It is possible to use blobs that are statistically concealing, though merely computationally binding, to get efficient statistical zero-knowledge computationally convincing protocols [Brassard 1991] (also known as *arguments* [Brassard and Crépeau 1990]). The blobs of Brassard and Crépeau [1986], which are based on the assumption that factoring is intractable, have all the properties we require.

Definition 8.1 (*Brassard-Crépeau* [1986]). Let P and Q be two primes of equal bit-length chosen by the Verifier and let α be a fixed random quadratic residue modulo $N = PQ$, also chosen by the Verifier. The Verifier announces N and α to the Prover. We define "SR-blobs" by $\mathscr{E}(0, x) = x^2 \pmod N$ and $\mathscr{E}(1, x) = \alpha x^2 \pmod N$, for $x \in \mathbb{Z}_N^*$. (Before this bit commitment scheme can be used, the Verifier must convince the Prover that α is a quadratic residue modulo N, which is why the resulting protocol is merely statistical rather than perfect zero-knowledge.)

It should be noted that, although QR-blobs (Section 2) can be used when the confidence level is a constant (because the blobs for the preprocessing stage can be published in advance), SR-blobs cannot be used in this case because the Verifier must produce the modulus N and the quadratic residue α. Furthermore, SR-blobs cannot be used in conjunction with the techniques of Sections 6 and 7 for similar reasons. On the other hand, Kilian's approach [1992] becomes significantly more interesting when applied to arguments.

It should also be noted that the unconditionally concealing blobs based on the certified discrete logarithm assumption [Boyar et al. 1990; Chaum et al. 1988] cannot be used in our context because they do not provide for the noninteractive processing of XOR gates. This is unfortunate since those blobs would otherwise allow for a *perfect* zero-knowledge computationally convincing protocol. Nevertheless, the discrete logarithm assumption can be used to implement yet another bit commitment scheme, which is suitable for our application [Boyar and Damgård 1990], but that scheme is merely statistically concealing.

9. *Related Results*

In Sections 4 and 5, the matrices are constructed such that matrix A contains only a_i's and zeros and matrix B contains only b_i's and zeros. This restriction is unnecessary. In fact, because of the noninteractive processing of XOR gates, the entries in both A and B could have the form $\sum_{j=1}^{k} x_j$ where each $x_j \in \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n, 1\}$, that is, these entries could be affine combinations of the variables. Allowing affine combinations is reasonable because the Prover and Verifier could still independently calculate blobs for the A and B matrices, whereas allowing more complicated functions of the variables

would probably require that the Prover first compute them and then show the Verifier that they were computed correctly. The following is an example of how affine combinations might be used, in 2×2 matrices:

$$\begin{pmatrix} a_1 + a_2 + b_1 + 1 & b_1 \\ a_1 + a_2 & a_1 \end{pmatrix} \times \begin{pmatrix} b_1 & b_2 \\ a_2 & b_2 \end{pmatrix}$$

$$= \begin{pmatrix} a_1 b_1 & a_1 b_2 + a_2 b_2 + b_2 \\ a_1 b_1 + a_2 b_1 + a_1 a_2 & a_2 b_2 \end{pmatrix}.$$

This example just gives $m = n = 2$, which is no improvement over what can be done without using affine combinations. In fact, there are no known examples where removing the original restrictions gives any improvement. For a given matrix size m, let $M(m)$ denote the maximum value that n can have. In Boyar et al. [1994], it is shown that $M(m) \leq m^2/3^{1/3} + O(m)$, even if affine combinations are allowed. If this many entries were possible asymptotically, this would significantly improve the results in this paper. It remains an open problem as to whether or not this is possible.

In attempting to extend this idea, one might consider also allowing the "good" elements in the C matrix, which are now restricted to being of the form $a_s b_s$, to have the form $a_s b_s + X$, where X is an affine combination of the a_i's and b_i's. For example, the term containing $a_1 b_1$ could be $a_1 b_1 + a_1 + b_3 + a_5 + 1$. Unfortunately, even if one could produce matrices in this form that worked, it would not lead to a general algorithm giving an improvement for all circuits (unless the X in $a_s b_s + X$ is restricted to containing no variables other than a_s and b_s). The problem that arises is that c_s is the output of the sth gate, so it is probably the input to some other gate. Thus, it is identical to at least one of the a_i's or b_i's. If some variable which is identical to c_s is included in X, then the matrix element in question would not depend on c_s. In the example, if c_1 was on the same wire as b_3, the resulting term would not depend on c_1. Thus, there would be no check of $a_1 b_1 = c_1$, and the Prover could cheat at that gate without the Verifier detecting it. This idea of allowing affine combinations in the C matrix could conceivably lead to good heuristics, however, which might work for many circuits. Nevertheless, there is a limit as to how much this could help, since the bounds from Boyar et al. [1994] can be extended to apply here.

One should keep in mind that the techniques given here are general in that they could be applied to any circuit. For any specific circuit, it might be possible to do significantly better, particularly if many of the variables are identical. For example, if the variables on the wires are $a_1, a_2, \ldots, a_m$, $b_{1,1}, b_{1,2}, \ldots, b_{m,m}$, and the outputs of the AND gates are $a_i b_{i,j}$ for $1 \leq i \leq m$ and $1 \leq j \leq m$, then these m^2 gates could be checked using $m \times m$ matrices, with the a_i's down the diagonal of the first matrix and the $b_{i,j}$'s placed in the obvious way in the second matrix.

After reading preliminary versions of this paper, several researchers have suggested nice improvements. We have already discussed Kilian's idea of using holographic proofs in order to reduce communication complexity of zero-knowledge proofs [Kilian 1992], Szemerédi's improvement on our original $O(n^{6/5} + kn^{3/5})$ result (in fact, Szemerédi was working on a graph problem, probably unaware of its implication on our technique), and Bellare's idea for simplifying the generation of cheap blobs.

Appendix A. Our Best Practical Scheme

Here is our *simple* efficient construction for processing 32 ANDs collectively by the techniques of Section 4: we put 32 entries in an 8×8 matrix as follows, where the asterisk is used to denote garbage terms.

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & 0 & 0 & 0 & 0 \\ a_5 & 0 & a_6 & 0 & a_7 & 0 & a_8 & 0 \\ a_9 & a_{10} & 0 & 0 & a_{11} & a_{12} & 0 & 0 \\ a_{13} & 0 & 0 & a_{14} & 0 & a_{15} & a_{16} & 0 \\ 0 & a_{17} & a_{18} & 0 & a_{19} & 0 & 0 & a_{20} \\ 0 & 0 & a_{21} & a_{22} & 0 & 0 & a_{23} & a_{24} \\ 0 & a_{25} & 0 & a_{26} & 0 & a_{27} & 0 & a_{28} \\ 0 & 0 & 0 & 0 & a_{29} & a_{30} & a_{31} & a_{32} \end{pmatrix} \times \begin{pmatrix} b_1 & 0 & 0 & 0 & b_{13} & b_5 & b_9 & 0 \\ 0 & b_2 & 0 & 0 & b_{25} & b_{17} & 0 & b_{10} \\ 0 & 0 & b_3 & 0 & b_{21} & 0 & b_{18} & b_6 \\ 0 & 0 & 0 & b_4 & 0 & b_{22} & b_{26} & b_{14} \\ b_{19} & b_7 & b_{11} & 0 & b_{29} & 0 & 0 & 0 \\ b_{27} & b_{15} & 0 & b_{12} & 0 & b_{30} & 0 & 0 \\ b_{23} & 0 & b_{16} & b_8 & 0 & 0 & b_{31} & 0 \\ 0 & b_{24} & b_{28} & b_{20} & 0 & 0 & 0 & b_{32} \end{pmatrix}$$

$$= \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & * & * & * & * \\ * & c_7 & * & c_8 & * & c_5 & * & c_6 \\ * & * & c_{11} & c_{12} & * & * & c_9 & c_{10} \\ * & c_{15} & c_{16} & * & c_{13} & * & * & c_{14} \\ c_{19} & * & * & c_{20} & * & c_{17} & c_{18} & * \\ c_{23} & c_{24} & * & * & c_{21} & c_{22} & * & * \\ c_{27} & * & c_{28} & * & c_{25} & * & c_{26} & * \\ * & * & * & * & c_{29} & c_{30} & c_{31} & c_{32} \end{pmatrix}$$

Recall that this construction can be used recursively to process $n = 32^t$ AND gates collectively, at the cost of $O(8^t)$ bit commitments per round. Although Szemerédi's technique (Section 5) improves on this construction from an asymptotic point of view, it is likely that the latter remains preferable for all reasonable values of n.

Appendix B. The Matrix Multiplication Protocol and Its Security

In Section 4, we reduced the problem of proving satisfiability of a Boolean circuit to the problem of verifying the product of two matrices over **GF**(2), given blob-based encryptions of the two matrices and of their product. Here, we give a complete and formal description of the protocol for matrix multiplication and prove that it is zero-knowledge.

For simplicity, we do not include in this appendix any of the several optimizations possible (such as those given in Section 6). We use the technique of Boyar et al. [1993] to demonstrate in zero-knowledge that $b_1 \wedge b_2 = b_3$ given blobs $F(b_1)$, $F(b_2)$, and $F(b_3)$. Let A, B, and C be three secret committed $m \times m$ matrices for which the Prover wants to convince the Verifier that $C = AB$. Let $F(A)$, $F(B)$ and $F(C)$ be blob-based encryptions of those matrices, which are available to the Verifier. The following protocol is used:

Protocol for matrix multiplication

(1) Repeat k times steps (2) through (8).
(2) The Verifier chooses m-bit vectors $\vec{x}$ and $\vec{y}$ uniformly at random and sends them to the Prover.

(3) The Prover and Verifier each independently compute the same blob-based encryptions $F(\vec{x} \cdot A)$, $F(B \cdot \vec{y})$ and $F(\vec{x} \cdot C \cdot \vec{y})$.
(4) For each i, let $z_i = (\vec{x} \cdot A)_i \wedge (B \cdot \vec{y})_i$ and let $u_i = F(z_i)$.
Let $\vec{z}$ be the vector $(z_1, z_2, \ldots, z_m)$.
The Prover sends $F(\vec{z}) = (u_1, u_2, \ldots, u_m)$.
(5) For each i the Prover sends, in random order, blobs $w_1^{(i)}$, $w_2^{(i)}$, and $w_3^{(i)}$ such that
(i) $w_1^{(i)}$ encodes 0.
(ii) $w_2^{(i)}$ encodes $(\vec{x} \cdot A)_i$.
(iii) $w_3^{(i)}$ encodes $(B \cdot \vec{y})_i$.
(6) The Verifier sends a random challenge $b = 0$ or $b = 1$.
(7) (i) If $b = 0$, the Prover shows, for each i, that $w_1^{(i)}$, $w_2^{(i)}$, and $w_3^{(i)}$ were created according to the rule given in step (5). More precisely, she tells him in which order she had sent $w_1^{(i)}$, $w_2^{(i)}$, and $w_3^{(i)}$, she opens $w_1^{(i)}$ and the (noninteractively computed) XORs of $w_2^{(i)}$ with $F((\vec{x} \cdot A)_i)$ and of $w_3^{(i)}$ with $F((B \cdot \vec{y})_i)$; all three blob openings should show the bit 0.
(ii) If $b = 1$, the Prover shows, for each i, that two of the $w_j^{(i)}$ blobs match u_i. Again, this is accomplished by showing that the XOR of the selected blobs with u_i are encryptions of zero.
(8) The Prover and the Verifier noninteractively compute the blob $F((\vec{x} \cdot C \cdot \vec{y}) \oplus (\oplus_{i=1}^{m} z_i))$; the Prover opens this blob to show it encodes a 0. □

It is immediate from the discussion in Section 3 that the protocol always succeeds if both parties follow it faithfully and if $C = AB$, whereas the Verifier will catch the Prover in the act of cheating with probability at least $1 - (7/8)^k$ if in fact $C \neq AB$, regardless of the Prover's cheating strategy and computing power. This is therefore a valid interactive proof system [Goldwasser et al. 1989] for committed matrix multiplication verification.

In order to prove that the protocol is zero-knowledge, we assume that the reader is familiar with the notion of a *Simulator* [Goldwasser et al. 1989]. For simplicity, we assume also that the Verifier does not "misbehave" (such as responding "2" when he is expected to say either "0" or "1", or, worse, not responding at all when it is his turn to communicate). It is an elementary exercise to also take account of misbehaving verifiers.

THEOREM B.1. *The matrix multiplication protocol is zero-knowledge when QR-blobs are used provided the Quadratic Residuosity Assumption holds.*

PROOF. We produce a Simulator. Note that the Simulator knows $F(A)$, $F(B)$, and $F(C)$, but not A, B, or C.

Simulator Algorithm

(1) Repeat k times steps (2) through (7).
(2) Receive $\vec{x}$ and $\vec{y}$ from the Verifier.
(3) Choose random values with Jacobi symbol $+1$ for the first $m - 1$ components of $F(\vec{z})$, and let u be their product. Choose a random value r, and let the last component of $F(\vec{z})$ be $r^2(u \cdot F(\vec{x} \cdot C \cdot \vec{y}))^{-1}$. Note that this will allow the Simulator to respond correctly at step (8) of the protocol. (Recall that the honest Prover would have chosen $z_i = (\vec{x} \cdot A)_i \wedge (B \cdot \vec{y})_i$.)
(4) Flip a fair coin to guess which challenge will be asked, and set up the $3m$ auxiliary blobs in order to answer that challenge. Thus, if the challenge guessed was $b = 0$, set up the auxiliary blobs so that one corresponds to $F((\vec{x} \cdot A)_i)$, one to $F((B \cdot \vec{y})_i)$, and one encrypts 0 for each i, $1 \leq i \leq m$; whereas if the challenge guessed was $b = 1$, set up the auxiliary blobs so that all three of them correspond to $F(z_i)$ for each i. To create a blob that corresponds to blob B, the Simulator chooses a random value r and computes r^2B^{-1}. This allows it to prove that the XOR of the two blobs encodes a 0.
(5) Receive the challenge b from the Verifier.

(6) If the actual challenge was not what the Simulator had guessed, the Simulator goes back to step (4) after rewinding the transcript and resetting the Verifier appropriately; otherwise, it continues with the next step.
(7) Finish this round off as the honest Prover would.

The distribution created by simulated transcripts is very different from the distribution that would occur with the true Prover, but, under the Quadratic Residuosity Assumption, no probabilistic polynomial-time Verifier would be able to detect the difference. Thus, the transcripts produced are computationally indistinguishable from those produced with the true Prover. Furthermore, the Simulator has very close to a 50% chance of correctly guessing the Verifier's challenge at each iteration of step (4), so the simulation is obtained in expected polynomial time. □

Appendix C. The Security of the Circuit Satisfiability Protocol

From the proof of Theorem B.1, it is straightforward to conclude that the full protocol for circuit satisfiability is also zero-knowledge. It is another matter altogether to prove that the protocol is a proof of knowledge. We assume that the reader is familiar with the notion of *Extractor* used to prove that a protocol is a proof of knowledge [Feige et al. 1988] (the term, *Extractor*, was not yet coined when [Feige et al. 1988] was written).

The subtlety involved in this issue is best illustrated by the fact that our matrix multiplication protocol itself does not necessarily allow an Extractor to determine the elements of the three matrices involved. In fact, it is not hard to verify that the information obtained by the Extractor is always compatible with the case $A = B = C = 0$, assuming the Quadratic Residuosity Assumption. However, the circuit satisfiability protocol is a proof of knowledge, in the sense that the Extractor can find a satisfying assignment to the circuit, though it may not be the same one the Prover used. So the Verifier obtains a "proof" that the Prover can satisfy the circuit.

THEOREM C.1. *When the security parameter $k = n$, the protocol of Section 4 for proving any circuit with sublinear bit commitments per round is a restricted input proof of knowledge, as defined by Feige et al.* [1988].

PROOF. The completeness is obvious, because if the Prover knows a satisfying assignment and she follows the protocol, the Verifier will accept.

To show soundness, we show how a probabilistic polynomial-time Extractor can obtain appropriate values for the elements of the matrices A, B, and C by playing the role of the Verifier. This is enough since those matrices contain the Boolean value associated with each wire in the circuit. Assume that the contents of the Prover's knowledge tape and random tape are such that the Verifier accepts with probability at least $1/p(n)$ for some polynomial p. Consider the computation tree associated with this protocol. Since the Verifier flips $2m + 1$ bits each round, this is a complete tree of degree 2^{2m+1}. An execution of the protocol corresponds to a root-to-leaf traversal on the tree. A node v in the tree is an "accepting" node if the Verifier accepts on each of the rounds on the path from the root to v. By assumption, the proportion of leaves which are accepting is at least $1/p(n)$. If we truncate the tree at nonaccepting nodes, there will be some level of the tree for which the Verifier accepts with probability at least $15/16$. (Otherwise, the proportion of accepting leaves in the

full tree is less than $(15/16)^n < 1/p(n)$.) The probability is taken over the contents of the portion RV of the Verifier's random tape which is used. This portion contains bits for the m-bit vectors $\vec{x}$ and $\vec{y}$ and the challenge bits b. In probabilistic polynomial time, the Extractor can use random sampling to estimate the probability of acceptance for each level in the tree, until it finds one with probability at least 15/16. Sampling within this level, the Extractor can find a node for which at least 7/8 of the $\vec{x}, \vec{y}$ pairs lead to the Prover being able to successfully answer both challenges. In the following, we assume that we are working with that node.

Extractor Algorithm

(1) Associate symbolic Boolean variables with the wires, one for each wire in the circuit $\mathscr{C}$. (The goal is to find values for these variables and thus find a satisfying assignment to the circuit $\mathscr{C}$.)
(2) Put these variables in some extra $m \times m$ matrices, $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$, exactly as the corresponding blobs are put in the A, B, and C matrices. Fill the remaining elements of $\tilde{A}$ and $\tilde{B}$ with zeros.
(3) Put distinct Boolean variables in the elements of $\tilde{C}$ which correspond to garbage elements in C.
(4) Start producing a set S of congruences:
   —If v is the variable on the output wire, add the congruence $\{v \equiv 1\}$ to S.
   —If v_{in} is the variable on the input to a NOT gate and v_{out} is the variable on its output, add the congruence $\{v_{out} \equiv v_{in} + 1\}$ to S.
(5) Try random vectors for $\vec{x}$ and $\vec{y}$ until enough (where "enough" is specified below) pairs $\vec{x}^{(j)}$, $\vec{y}^{(j)}$ are found such that for both the challenge $b = 0$ and the challenge $b = 1$, the Prover is able to successfully answer the challenge.
(6) For every pair $\vec{x}^{(j)}$, $\vec{y}^{(j)}$, introduce new variables $z_1^{(j)}, z_2^{(j)}, \ldots, z_m^{(j)}$ for the outputs of the AND gates and a variable $z^{(j)}$ for $\vec{x}^{(j)} \cdot C \cdot \vec{y}^{(j)}$.
   —Add the congruence $\{z^{(j)} \equiv \sum_{i=1}^{m} z_i^{(j)}\}$ to S.
   —Let F be the affine formula for $\vec{x}^{(j)} \cdot \tilde{C} \cdot \vec{y}^{(j)}$, in terms of the variables in $\tilde{C}$. Add the congruence $\{z^{(j)} \equiv F\}$ to S.
   —The answers to the challenge bits $b = 0$ and $b = 1$ together show that each $z_i^{(j)}$ is equal to one of the inputs to the corresponding AND gate and either equal to zero or also equal to the other input. Add to S the congruences implied by this. (At this point, one could substitute affine combinations for the $z_i^{(j)}$'s and eliminate them from the set of congruences.)
(7) Solve the system of equations for the variables in $\tilde{A}$ and $\tilde{B}$, thereby obtaining a satisfying assignment for the circuit $\mathscr{C}$.

The number of $\vec{x}^{(j)}, \vec{y}^{(j)}$ pairs required in step (5) is $O(m^2)$. To see this, for each $\vec{x}^{(j)}, \vec{y}^{(j)}$ pair, create a vector $\vec{u}^{(j)}$ of length m^2. This vector is defined below:

$$U(\vec{x}, \vec{y})_{(k-1)m+l} = \begin{cases} 1 & \text{if } x_k = y_l = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$u^{(j)}_{(k-1)m+l} = U(\vec{x}^{(j)}, \vec{y}^{(j)})_{(k-1)m+l}.$$

If the set of $\vec{u}^{(j)}$ vectors corresponding to the $\vec{x}^{(j)}, \vec{y}^{(j)}$ pairs is a basis for the vector space $[\mathbf{GF}(2)]^{m^2}$, then each variable in the $\tilde{C}$ matrix has been selected and every necessary condition relating to it has been added to S. To see this, consider the variable in $\tilde{C}[k, l]$. Since the set of $\vec{u}^{(j)}$ vectors forms a basis, there is a subset $U_{k,l}$ which, when added together gives the vector which is all zeros, except for a one in location $(k-1)m + l$. The vectors in $U_{k,l}$ came from a set $V_{k,l}$ of $(\vec{x}, \vec{y})$ vector pairs. The sum $\sum_{(\vec{x},\vec{y}) \in V_{k,l}} \vec{x} \cdot \tilde{C} \cdot \vec{y}$ is then $\tilde{C}[k, l]$, since

the variables in $\tilde{C}$ are also over **GF**(2). All of the congruences for the pairs in $V_{k,l}$ have obviously been added to S. These congruences can simply be added to give the appropriate congruences for determining $\tilde{C}[k, l]$.

Assuming that N is a Blum integer, the Extractor can solve the system of congruences in S for the variables in the matrices $\tilde{A}$ and $\tilde{B}$ and thus obtain a satisfying assignment. This can be done because the Extractor now has the same information as the Extractor in Boyar et al. [1993, Theorems 6 and 8], so it can use the techniques developed there to find a satisfying assignment for the circuit. The initialization subprotocol referred to in Definition 2.2 ensures that the probability that N is not a Blum integer is negligible.

All that is left to show is that this Extractor is probabilistic polynomial time. To see this, it is necessary to show that the Extractor can find enough $\vec{x}^{(j)}, \vec{y}^{(j)}$ pairs. It can do this because it can choose the pairs $\vec{x}, \vec{y}$ at random to create a set $\mathscr{V}$ of $\vec{u}$ vectors that are linearly independent. It is shown below that whenever the vectors in $\mathscr{V}$ fail to span $[\mathbf{GF}(2)]^{m^2}$, choosing a pair of $\vec{x}, \vec{y}$ vectors at random will produce a $\vec{u}$ vector outside the subspace spanned by $\mathscr{V}$ with probability at least $1/4$. Since the Prover is able to answer both challenges for at least $7/8$ of the $\vec{x}, \vec{y}$ pairs, a sufficient set of vector pairs can be found in probabilistic polynomial time.

CLAIM C.2. *Suppose T is a subspace of $[\mathbf{GF}(2)]^{m^2}$ of dimension $\leq m^2 - 1$. Then there are at least 2^{2m-2} pairs, $\vec{x}, \vec{y}$, for which $\vec{u} = U(\vec{x}, \vec{y})$ is outside of T.*

PROOF. Suppose T has dimension $d \leq m^2 - 1$. There exists a basis B for T such that when the d vectors of B are placed in the rows of a $d \times m^2$ matrix M (possibly with a renumbering of the columns), the first d columns of M form an identity matrix. Thus if $\vec{v}^{(i)}$ is the ith vector in B, then $v_i^{(i)} = 1$, and for all $w \in B$, if $\vec{w} \neq \vec{v}^{(i)}$, then $w_i = 0$.

Suppose there is some column j of M that contains only zeros. Express j as $(k-1)m + l$, where $k, l \in \{1,..,m\}$. Then for any $\vec{x}, \vec{y}$ pair such that $x_k = y_l = 1$, the vector $\vec{u} = U(\vec{x}, \vec{y})$ is outside T. There are 2^{2m-2} vector pairs that have this property.

Suppose there is no such column j. Then there is a vector, $\vec{v}^{(i)} \in B$, that has a one in at least one component $j > d$, in addition to having a one in component i. For any vector pair $\vec{x}, \vec{y} \in [\mathbf{GF}(2)]^m$, if $U(\vec{x}, \vec{y}) \in T$, then changing at most two bits of the original pair will give a new pair $\vec{x}', \vec{y}'$ such that $U(\vec{x}', \vec{y}') \notin T$. These bits are $x_{\lceil i/m \rceil}$ and $y_{f(i,m)}$, where

$$f(i, m) = \begin{cases} i \bmod m & \text{if } (i \bmod m) \neq 0 \\ m & \text{otherwise.} \end{cases}$$

Let $\vec{x}'$ denote the vector $\vec{x}$ with bit $x_{\lceil i/m \rceil}$ complemented, and let $\vec{y}'$ denote the vector $\vec{y}$ with bit $y_{f(i,m)}$ complemented. Suppose $\{U(\vec{x}, \vec{y}), U(\vec{x}, \vec{y}'), U(\vec{x}', \vec{y}), U(\vec{x}', \vec{y}')\} \subseteq T$. Then

$$\vec{v}' = U(\vec{x}, \vec{y}) + U(\vec{x}, \vec{y}') + U(\vec{x}', \vec{y}) + U(\vec{x}', \vec{y}') \in T$$

and it contains all zeros except for the ith component, which is a one. This is a contradiction, since $\vec{v}^{(i)}$ is the only vector in T that has a one in component i and zeros in all of the other first d components. Therefore, at least one of the four vectors in $\{U(\vec{x}, \vec{y}), U(\vec{x}, \vec{y}'), U(\vec{x}', \vec{y}), U(\vec{x}', \vec{y}')\}$ must not be in T. Thus, at least $1/4$ of the $\vec{x}, \vec{y}$ pairs are such that $U(\vec{x}, \vec{y}) \notin T$. □

Thus, in probabilistic polynomial time, the Extractor can produce a set of values for the variables in $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$. Since the vector pairs $(\vec{x}^{(j)}, \vec{y}^{(j)})$ were chosen to select every element in $\tilde{C}$, if $\tilde{C}[k,l]$ is an element in $\tilde{C}$ that represents a value on a wire of the circuit, the values in $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ must be such that $\tilde{C}[k,l] = \tilde{A}[k,j] \cdot \tilde{B}[j,l]$ for the appropriate j. Hence, we can assign the values of these variables to the corresponding wires in the circuit to obtain a consistent computation and a satisfying assignment. □

ACKNOWLEDGMENTS. We are grateful to Mihir Bellare, Lance Fortnow, Joe Kilian, Kim Larsen, Silvio Micali and Steven Rudich for their interest and suggestions. We also acknowledge Sampath Kannan, Steven Rudich and Gábor Tardos for their role in our finding out about Szemerédi's technique. Moreover, Vašek Chvátal, Geňa Hahn, Sophie Laplante and Bill Steiger were very helpful in our struggle with the function ν.

REFERENCES

ALON, N., GOLDREICH, O., HÅSTAD, J., AND PERALTA, R. 1992. Simple constructions of almost k-wise independent random variables. *Ran. Struct. Algorithms 3*, 3, 289–304.

ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. 1992. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*. (Pittsburgh, Pa., Oct. 24–27). IEEE Computer Society Press, Los Alamitos, Calif., pp. 14–23.

BABAI, L., FORTNOW, L., LEVIN, L. A., AND SZEGEDY, M. 1991. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (New Orleans, La., May 6–8). ACM, New York, pp. 21–31.

BACH, E. 1991. Realistic analysis of some randomized algorithms. *J. Comput. Syst. Sci. 42*, 1, 30–53.

BLUM, M. 1987. How to prove a theorem so that no one else can claim it. In *Proceedings of the 1986 International Congress of Mathematicians* (Berkeley, Calif., Aug. 3–11). American Mathematical Society, Providence, R.I., pp. 1444–1451.

BLUM, M., FELDMAN, P., AND MICALI, S. 1988. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, Ill., May 2–4). ACM, New York, pp. 103–112.

BLUM, M., AND MICALI, S. 1984. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput. 13*, 850–864.

BOYAR, J., AND DAMGÅRD, I. B. 1990. A discrete logarithm blob for noninteractive XOR gates. Tech. Rep. DAIMI PB-327 (Aug.). Computer Science Dept., Aarhus Univ., Denmark.

BOYAR, J., FICH, F., AND LARSEN, K. 1994. Bounds on certain multiplications of affine combinations. *Disc. Appl. Math. 52*, 155–167.

BOYAR, J., KRENTEL, M., AND KURTZ, S. 1990. A discrete logarithm implementation of perfect zero-knowledge blobs. *J. Crypt. 2*, 2, 63–76.

BOYAR, J., LUND, C., AND PERALTA, R. 1993. On the communication complexity of zero-knowledge proofs. *J. Crypt. 6*, 2, 65–85.

BOYAR, J., AND PERALTA, R. 1994. Efficient non-interactive zero-knowledge proofs of circuit satisfiability. Preprint No. 1, Institut for Matematik og Datalogi. Odense Univ., Denmark.

BRASSARD, G. 1991. Cryptology column—How convincing is your protocol? *ACM SIGACT News 22*, 1, 5–12.

BRASSARD, G., CHAUM, D., AND CRÉPEAU, C. 1988. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci. 37*, 156–189.

BRASSARD, G., AND CRÉPEAU, C. 1986. Non-transitive transfer of confidence: A *perfect* zero-knowledge interactive protocol for SAT and beyond. In *Proceedings of the 27th IEEE Annual Symposium on Foundations of Computer Science*. (Toronto, Ontario, Canada, Oct. 27–29). IEEE Computer Society Press, Los Alamitos, Calif., pp. 188–195.

BRASSARD, G., AND CRÉPEAU, C. 1987. Zero-knowledge simulation of Boolean circuits. In *Advances in Cryptology—Proceedings of Crypto '86* (Santa Barbara, Calif., August 11–15, 1986). Springer-Verlag, Berlin, Germany, pp. 223–233.

BRASSARD, G., AND CRÉPEAU, C. 1990. Sorting out zero-knowledge. In *Advances in Cryptology —Proceedings of Eurocrypt '89* (Houthalen, Belgium, April 10–13, 1989). Springer-Verlag, Berlin, Germany, pp. 181–191.

CHAUM, D. 1987. Demonstrating that a public predicate can be satisfied without revealing any information about how. In *Advances in Cryptology—Proceedings of Crypto '86* (Santa Barbara, Calif., August 11–15, 1986). Springer-Verlag, Berlin, Germany, pp. 195–199.

CHAUM, D., DÅMGARD, I. B., AND VAN DE GRAAF, J. 1988. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology—Proceedings of Crypto '87* (Santa Barbara, Calif., August 16–20, 1987). Springer-Verlag, Berlin, Germany, pp. 87–119.

COPPERSMITH, D., AND WINOGRAD, S. 1990. Matrix multiplication via arithmetic progressions. *J. Symb. Comput. 9*, 251–280.

DÅMGARD, I. B. 1990. On the randomness of Legendre and Jacobi sequences. In *Advances in Cryptology—Proceedings of Crypto '88* (Santa Barbara, Calif., August 21–25, 1988). Springer-Verlag, Berlin, Germany, pp. 163–172.

DÅMGARD, I. B. 1993. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *Advances in Cryptology—Proceedings of Eurocrypt '92* (Balatonfured, Hungary, May 24–28, 1992). Springer-Verlag, Berlin, Germany, pp. 341–355.

DE SANTIS, A., MICALI, S., AND PERSIANO, G. 1988. Non-interactive zero-knowledge proof systems. In *Advances in Cryptology—Proceedings of Crypto '87* (Santa Barbara, Calif., August 16–20, 1987). Springer-Verlag, Berlin, Germany, pp. 52–72.

FEIGE, U., FIAT, A., AND SHAMIR, A. 1988. Zero-knowledge proofs of identity. *J. Crypt. 1*, 2, 77–94.

FEIGE, U., LAPIDOT, D., AND SHAMIR, A. 1990. Multiple non-interactive zero-knowledge proofs based on a single random string. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science* (St. Louis, Missouri, Oct. 22–24). IEEE Computer Society Press, Los Alamitos, Calif., pp. 308–317.

FREIVALDS, R. 1979. Fast probabilistic algorithms. In *Proceedings of the 8th Symposium on the Mathematical Foundations of Computer Science* (Olomouc, Czechoslovakia, Sept. 3–7). Springer-Verlag, Berlin, Germany, pp. 57–69.

GOLDREICH, O., MICALI, S., AND WIGDERSON, A. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM 38*, 3 (July), 691–729.

GOLDWASSER, S., AND MICALI, S. 1984. Probabilistic encryption. *J. Comput. Syst. Sci. 28*, 270–299.

GOLDWASSER, S., MICALI, S., AND RACKOFF, C. 1989. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 186–208.

KILIAN, J. 1992. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing* (Victoria, B.C., Canada, May 4–6). ACM, New York, pp. 723–732.

KILIAN, J. 1994. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science* (Santa Fe, New Mexico, Nov. 20–22). IEEE Computer Society Press, Los Alamitos, Calif., pp. 466–477.

KILIAN, J., MICALI, S., AND OSTROVSKY, R. 1989. Minimum resource zero-knowledge proofs. In *Proceedings of the 30th IEEE Annual Symposium on Foundations of Computer Science* (Research Triangle Park, North Carolina, Oct. 30–Nov. 1). IEEE Computer Society Press, Los Alamitos, Calif., pp. 474–479.

MOSER, L. 1953. On non-averaging sets of integers. *Can. J. Math. 5*, 245–252.

NAOR, J., AND NAOR, M. 1990. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing* (Baltimore, Md., May 12–14). ACM, New York, pp. 213–223.

NAOR, M. 1991. Bit commitment using pseudo-randomness. *J. Crypt. 4*, 2, 151–158.

PERALTA, R. 1990. On the randomness complexity of algorithms. Computer Science Res. Rep. TR 90-1. Univ. of Wisconsin–Milwaukee, Milwaukee, Wis.

PERALTA, R. 1992. On the distribution of quadratic residues and non-residues modulo a prime number. *Math. Comput. 58*, 433–440.

SALEM, R., AND SPENCER, D. C. 1942. On sets of integers which contain no three terms in arithmetical progression. *Proc. Nat. Acad. Sci. U.S.A. 28*, 561–563.

VAN DE GRAAF, J., AND PERALTA, R. 1988. A simple and secure way to show the validity of your public key. In *Advances in Cryptology—Proceedings of Crypto '87* (Santa Barbara, Calif., August 16–20, 1987). Springer-Verlag, Berlin, Germany, pp. 128–134.

YAO, A. 1982. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Annual Symposium on Foundations of Computer Science* (Chicago, Ill., Nov. 3–5). IEEE Computer Society Press, Los Alamitos, Calif., pp. 80–91.

RECEIVED APRIL 1993; REVISED JULY 1994; ACCEPTED MAY 1995