A note on efficient zero-knowledge proofs and arguments.

(extended abstract)

Joe Kilian
NEC Research Institute
Princeton, NJ 08540

Abstract

In this note, we present new zero-knowledge interactive proofs and arguments for languages in NP. To show that $x \in L$, with an error probability of at most 2^{-k}, our zero-knowledge proof system requires $O(|x|^{c_1}) + O(\lg^{c_2}|x|)k$ ideal bit commitments, where c_1 and c_2 depend only on L. This construction is the first in the ideal bit commitment model that achieves large values of k more efficiently than by running k independent iterations of the base interactive proof system. Under suitable complexity assumptions, we exhibit a zero-knowledge arguments that require $O(\lg^c|x|)kl$ bits of communication, where c depends only on L, and l is the security parameter for the prover.[1] This is the first construction in which the total amount of communication can be less than that needed to transmit the NP witness. Our protocols are based on efficiently checkable proofs for NP [4].

[1] Typically, l is the size of some problem the poly-time bounded prover is assumed to be unable to solve.

1 Introduction.

1.1 The problem of efficient security amplification.

The standard definition of interactive proofs[7] requires that the verifier accept a correct proof and reject an incorrect assertion with probability at least $\frac{2}{3}$. As there are few applications where a 1/3 error probability is acceptable, one usually tries to obtain an error probability less than 2^{-k}, where k is some easily adjustable security parameter. The most obvious way of achieving this security amplification is take a protocol with a 1/3 error probability, run it $O(k)$ times, and have the verifier accept or reject by majority vote.[2] Are there any more efficient ways of achieving security than by this simple technique? As we will show, the answer is yes, for a wide variety of languages, in a well known model for which no other amplification technique was previously known.

The work of Boyar, Brassard and Peralta.

Our work is inspired by Boyar, Brassard and Peralta's zero-knowledge protocol for *circuit satisfiability*,[3] which for the first time achieved a more efficient security amplification than by the naive approach [1]. Their protocol achieved a 2^{-k} error probability using only $O(n^{1+\epsilon_n}+\sqrt{n}^{1+\epsilon_n})$ "ex-orable" bit commitments, where ϵ_n approaches 0 as n, the circuit size, approaches infinity. By an "ex-orable" bit commitment, we mean one in which the verifier can take the commitments for bits $b_1, \ldots, b_n$, and by himself generate a commitment

[2] In proof systems where the verifier always accepts a correct proof, it suffices to have the modified verifier accept iff he accepted in every run of the protocol.

[3] That is, given a circuit C, with a single output bit, is there an input on which C will output a 1?

for $b_1 \oplus \cdots \oplus b_n$. Such a bit commitment scheme may be based on the (unproven) intractability of computing quadratic residuosity modulo Blum integers [2].

Thus, after a somewhat expensive initialization process, the cost of achieving a 2^{-k} error probability is only $O(\sqrt{n}^{1+\epsilon_n} k)$ for very large k. In contrast, running the most efficient known zero-knowledge proof for circuit satisfiability (e.g., [3]) k times requires $\Omega(nk)$ bit commitments.

1.2 The results of this paper.

The work mentioned above leaves open three natural questions that we will address in this paper. First, what can be accomplished given an ideal bit commitment primitive, without the extra exclusive-or feature on which the [1] protocol heavily relies? Second, can one asymptotically achieve efficient amplification for arbitrary NP languages? While any NP language can be reduced to circuit satisfiability, the resulting size blow-up may eliminate the efficiency improvement. Finally, can one eliminate the expensive initialization step involved in these asymptotically efficient protocols? Because of this cost, the above technique cannot compete with the naive approach until k is at least $\Omega(n^{\epsilon_n})$.

Our first theorem shows that all NP languages have asymptotically efficient zero-knowledge proofs. By asymptotic efficiency, we mean the behavior of the protocol as k approaches infinity, in which case its initial start-up cost becomes negligible.

Theorem: Let ϵ be an arbitrary positive constant. In the ideal bit commitment model, there exists a zero-knowledge protocol for the circuit satisfiability problem that requires

$$O(n^{1+\epsilon} + (\lg^{O(1/\epsilon)} n)k)$$

bit commitments.

Since $\lg^{c_1} n^{c_2} = c_2^{c_1} \lg^{c_1} n$, we obtain asymptotically efficient protocols for any NP language. Unfortunately, we cannot eliminate the large start-up cost incurred by our protocol in the ideal bit commitment model. However, under certain complexity assumptions, we can transform our interactive proofs into ***arguments***, first developed by Brassard and Crépeau.[4] [2] that require very little total communication. Our result is as follows.

Theorem: Suppose there exists a perfect cryptographic bit committal scheme (In the argument model), parameterized by a security parameter l, and a family of collision intractable hash functions $\{F_{l,r} : \Sigma^{2l} \longrightarrow \Sigma^{l}$. Then for some constant c there exists an argument for the circuit satisfiability problem that requires $O((\lg^c n)l)$ bits of communication to achieve an error probability of $1/3$.

[4]Arguments are interactive proofs that involve a complexity assumption on the power of the prover. An additional security parameter, l, governs the size of the problems that the prover is assumed unable to solve. For example, the prover may be assumed to be unable to factor a product of 2 random l-bit primes.

Note that the smallest known NP witnesses for the circuit satisfiability problem are $\Omega(n)$, and for all NP-complete languages, the smallest known witness are of size $\Omega(n^\epsilon)$, for some $\epsilon > 0$. Indeed, if this were not the case for some NP-complete language, then

$$NP \in \bigcap_{\epsilon>0} DTIME(2^{n^\epsilon}),$$

which would be unexpected. Thus, arguments may be more concise than the most efficient NP proofs for extremely large problems, and reasonable security parameters.

1.3 Techniques used.

The results in this paper follow quite easily from three techniques. First, we use the notion of zero-knowledge proofs on committed bits, sometimes refer to as "notarized envelopes." Second, we use the "transparent proofs" of [4]. Finally, the efficient argument we implement uses collision-intractable hash functions as part of its committal protocol.

Notarized envelopes.

Notarized envelopes allow the prover to commit to a set of bits $b_1, \ldots, b_n$, and at some later point in time prove that some predicate $P(b_1, \ldots, b_n)$ holds for these bits, without revealing anything information about these committed values. Often in zero-knowledge proofs, the verifier asks the prover to reveal some set of bits, and then determines if he is happy with their revealed values. We can often replace this step by having the prover simply prove to the verifier that he would have been happy had he seen the revealed values. Since so little information is revealed to the verifier, it is possible to reuse much of the old proof without compromising the overall security of the proof. We will illustrate this example by showing how to increase the efficiency of Goldreich, Micali and Wigderson's zero-knowledge proof for graph 3-colorability [8].

Goldreich, Micali and Wigderson [9] based notarized envelopes (under a different name) on a complexity assumption. More recent constructions implement notarized envelopes based on ideal commitment schemes (envelopes). To make this note more self contained, we describe a scheme based on some unpublished work of Rudich and Bennett, and on the work of Brassard and Crépeau [2].

Transparent Proofs.

The main driving power behind this work is a deep theorem about NP. Babai, Fortnow, Levin and Szegedy have shown that any NP statement has a polynomial sized witness that may be checked in polylogarithmic time after some initial preprocessing on the statement. The transparent proofs presented by [4] are somewhat larger than the ordinary witnesses. For the circuit satisfiability problem, the transparent proof may be made to be of size $O(n^{1+\epsilon})$, for any $\epsilon > 0$, where n is the size of the circuit. However, the time needed to check their proof will be $\lg^{O(1/\epsilon)} n$, and this tradeoff will manifest itself in our work as a tradeoff between the initial work required for our zero-knowledge proof system and its asymptotic efficiency.

Collision-intractable hash functions.

In our zero-knowledge interactive proof system, the initial start-up cost is incurred when the prover must commit to all the bits of a large transparent proof. Using collision-intractable hash functions, it is possible to hash these bits down to a manageable number, yet still be able to efficiently reveal individual bits when necessary. We note that our trick only works if the prover is polynomially time-bounded, and we make some unproven complexity theoretic assumptions.

1.4 Outline of the paper.

The rest of this paper is outlines as follows. In Section 2, we outline an implementation of notarized envelopes based on an ideal bit commitment primitive. and illustrate our approach be speeding up a zero-knowledge proof system for graph 3-colorability. In Section 3, we exhibit an asymptotically efficient proof system for circuit satisfiability, based on an ideal bit commitment primitive. In Section 4, we exhibit an efficient argument for circuit satisfiability, based on suitable complexity assumptions.

2 Notarized envelopes.

In this section, we outline a well-known, though apparently unpublished scheme for notarized envelopes. Brassard and Crépeau essentially reduced notarized envelopes to envelopes with zero-knowledge proofs of equality assertions. In turn, researchers such as Bennett and Rudich have implemented these equality proofs by representing bits by "pair blobs," which we will describe below.

2.1 Using pair blobs to prove equality assertions.

The key to proofs of equality is to represent each bit as a random exclusive-or of two bits. With this representation, the commit and reveal operations are straightforward extensions to the old schemes.

COMMIT(x_i) P uniformly chooses $x_i^0, x_i^1 \in \{0,1\}$, subject to $x_i = x_i^0 \oplus x_i^1$, and commits to x_i^0 and x_i^1 using the ideal committal scheme.

REVEAL(x_i) The prover reveals x_i^0 and x_i^1 using the ideal committal scheme. V computes $x_i = x_i^0 \oplus x_i^1$.

We say that the value of a pair blob is the exclusive-or of its two bits. Given the above representation scheme, the following protocol allows one to prove that two committed bits are equal, without revealing their value. However, this scheme will only allow a committed bit to be involved in a single proof.

```
PROVE-EQUAL-NAIVE($x_i, x_j$) /* Prove that $x_i = x_j$ */
    1. $P$ sends $V$ the value of $x_i^0 \oplus x_j^0$.
    2. $V$ uniformly chooses $b \in \{0,1\}$, and sends $b$ to $P$.
    3. $P$ reveals $x_i^b$ and $x_j^b$, who accepts iff $x_i^b \oplus x_j^b$ is
       equal to the value sent in Step 1.
```

In order to use bits in more than one proof, the prover simply makes spare copies. The verifier either checks that the copies are equal to the original, using PROVE-EQUAL-NAIVE, or checks that the two originals are equal. In either case, there will be at least one "live" copy of each committed bit that may be used later in the protocol.

```
PROVE-EQUAL($x_i, x_j$) /* Prove that $x_i = x_j$,
                          nondestructively */
    1. $P$ commits to $x_i', x_i'', x_j', x_j''$, where each copy is
       equal to the original.
    2. With equal probability, $V$ asks $P$ to show, using
       PROVE-EQUAL, that
           A. $x_i = x_i'$ and $x_j = x_j'$, and
           B. $x_i = x_i''$ and $x_j = x_j''$, or
           C. $x_i = x_j$.
       $V$ rejects iff he rejects in the above proofs. The
       new representation for $(x_i, x_j)$ is taken to be
       $(x_i'', x_j'')$ in Case A, and $(x_i', x_j')$ in Cases B and C.
```

In our protocols, we will want to maintain the secrecy of the pair-blob representations of our bits as much as possible. We want

Definition 1 Let $B_1, \ldots, B_n$ be a set of pair blobs, and let $\hat{V}$ be some arbitrary verifier. We say that $B_1, \ldots, B_n$ is *secure* against $\hat{V}$ iff for any $(b_1^0,$ (b_n^0, b_n^1) and $(b_1'^0, b_1'^1), \ldots, (b_n'^0, b_n'^1)$ such that $b_i^0 \oplus b_i^1 = b_i'^0 \oplus b_i'^1$ for all i, we have that

$prob(B_1, \ldots, B_n = (b_1^0, b_1^1), \ldots, (b_n^0, b_n^1) | \hat{V}'s \text{ view})$ and,
$prob(B_1, \ldots, B_n = (b_1'^0, b_1'^1), \ldots, (b_n'^0, b_n'^1) | \hat{V}'s \text{ view})$

are the same.

Informally, $B_1, \ldots, B_n$ are secure against $\hat{V}$ if $\hat{V}$ has learned nothing about their internal representation, as opposed to knowing about their values. For example, $\hat{V}$ may know that B_1 represents a 0, but cannot tell whether $B_1 = (0,0)$ or $B_1 = (1,1)$.

The following lemmas can be proven using a finite case analysis. Lemma 1 says that if P behaves properly in protocol PROVE-EQUAL, then V will accept. Lemma 2 says that no verifier can gain extra information about the values of x_i and x_j, or about their new pair-blob representations. Lemma 3 says that if $x_i \neq x_j$, then V will reject with some constant nonzero probability. Finally, Lemma 4 says that no prover can try to alter the committed values of x_i and x_j in the protocol without being detected with some constant nonzero probability.

Lemma 1 If $x_i = x_j$, and P obeys the protocol, then V will always accept in protocol PROVE-EQUAL(x_i, x_j). The new pair blobs for x_i and x_j will have the same value as the original.

Lemma 2 If $x_i = x_j$, then the view of any verifier $\hat{V}$, will be independent of the actual values of x_i and x_j. If the pair blobs for x_i and x_j are secure against $\hat{V}$ at the beginning of the protocol, then they will be secure against $\hat{V}$ at the end of the protocol.

Lemma 3 If $x_i \neq x_j$, then with probability at least 1/6, V will reject, regardless of the strategy used by the prover.

Lemma 4 If $x_i = x_j$, then regardless of the strategy used by a (possibly malicious) prover $\hat{P}$, one of the following cases must hold at the conclusion of Step 1 of protocol PROVE-EQUAL:

1. V is guaranteed to reject with probability at least 1/6, or
2. The new pair blobs for x_i and x_j are guaranteed to have the same value as the originals. ∎

2.2 Proving general predicates.

Given "primitives" for proving equality for committed bits, we can now use, for example, the protocol of Brassard and Crépeau. We note that while a malicious prover can make false assertions, or even alter the value of some committed bits, this affects the soundness of their proof system by only a constant factor. A similar phenomenon has been noted in the study of other proof systems (c.f. [11]). We state, without proof, the resulting theorem one obtains.a

Theorem 1 [oral-tradition,[2]] There exists a protocol PROVE-CIRCUIT$(x_1, \ldots, x_k, C)$, where $x_1, \ldots, x_k$ have been committed to as random pair-blobs, and C is a circuit with n gates, such that the following properties hold:

1. The protocol requires at most $O(n)$ commitments/revelations and bits of communication.
2. If $C(x_1, \ldots, x_k) = 1$, and P obeys the protocol, then
   - V will always accept,
   - If the pair blobs for $x_1, \ldots, x_k$ were secure against $\hat{V}$ at the beginning of the protocol, the (possible different) pair blobs for $x_1, \ldots, x_k$ will be secure against $\hat{V}$ at the end of the protocol.
   - There exists an expected polynomial-time simulator $S(\hat{V}, C)$ that can simulate $\hat{V}$, just by having $\hat{V}$ as an oracle (we will say more about this simulator shortly).
3. If $C(x_1, \ldots, x_k) \neq 1$, then V will reject with probability $\frac{1}{12}$.

Here, the 1/12 factor is an artifact of our equality-proving protocol; it is easy to obtain any constant one wishes using only a constant number of envelopes.

Properties of the PROVE-CIRCUIT simulator.

In order for our own simulator constructions to work, we need to use special properties of the PROVE-CIRCUIT simulator. In the ideal bit commitment model, it is not unreasonably to allow the simulator to wait until it must reveal a bit before actually deciding on its value. However, to accommodate certain complexity based protocols, we must require the simulator to decide on the value of a committed bit at the same time it is committed. Committed bits may only be changed by saving the global state of the simulation (including $\hat{V}$'s state) before the commitment is made, and then later restoring this state. Our proofs' committed bits persist through many iterations of the protocol, so we do not allow our PROVE-CIRCUIT simulator to tamper with these commitments at all, even by global save/restore operations.

Fortunately, one can easily construct a simulator S for the PROVE-CIRCUIT protocol with the following properties:

Immutability: We assume that bits $x_1, \ldots, x_k$ are all 0, and that their pair-blob representations are secure against $\hat{V}$. S will not change any of these commitments. Note, however, that the pair blob used to represent x_i may be completely replaced if it is used in the PROVE-EQUAL protocol.

Genuine Commitments: S does not change the value of a bit once it has been committed, though it may "back up" the simulation to a time before this commitment took place.

Black-Box Simulation: Given any verifier $\hat{V}$, S can perfectly simulate the conversation $(P, \hat{V})$ by simply interacting with $\hat{V}$, but may choose to save and restore $\hat{V}$'s state. S will run in expected polynomial time, not counting the time used by $\hat{V}$. For the simulation to be correct, we require that the pair-blobs for the actual $x_1, \ldots, x_n$ in the protocol are initially secure against $\hat{V}$, and the pair blobs (all with value 0) used by the simulator are also initially secure against $\hat{V}$.

Security: If the pair blobs for $x_1, \ldots, x_k$ are secure against $\hat{V}$ at the beginning of the simulation, the (possibly different) pair blobs for $x_1, \ldots, x_k$ will be secure against $\hat{V}$ at the end of the simulation. Furthermore, the values of these pair-blobs will always be 0.

By the immutability property, S makes do with the faked commitments given to it, and does not try to change them. By the security property, S leaves the "faked" commitments for $x_1, \ldots, x_k$ in the same state as at the beginning of the protocol (i.e., uniformly committed 0's). These properties allow us to chain together a series of PROVE-CIRCUIT protocols.

3 Using notarized envelopes to make protocols more efficient.

In this section, we show that using notarized envelopes can increase the efficiency of interactive proof systems, and combine particularly well with transparent proofs. We first show how to use notarized envelopes to dramatically increase the efficiency of a well-known protocol for graph 3-colorability. While of no theoretical value, this example illustrate the basic idea behind this note. We next observe that if one uses the same technique with transparent proofs, one obtains bounds that are substantially better than any previously known.

3.1 Speeding up a zero-knowledge proof for 3-colorability.

We consider Goldreich, Micali and Wigderson's zero-knowledge proof system for graph 3-colorability. Their proof system went as follows.

1. P commits to a randomly permuted coloring χ of the graph G.
2. V randomly chooses an edge (i, j) of G, and sends (i, j) to P.
3. P reveals $\chi(i)$ and $\chi(j)$. V rejects if these two colors are different.
4. To achieve a 2^{-k} error probability, P and V run Steps 1–3 $O(mk)$ times.

If G is 3-colorable, and P behaves properly, then V will always accept. If G is not three-colorable, then V will choose a bad edge with probability at least $1/m$, where m is the number of G's edges. The protocol is zero-knowledge, since only one edge is revealed, and χ is a randomly permuted.

Now, revealing $\chi(k)$, for an additional node k, will compromise the security of the proof. Hence, all of these $O(n)$ bit committals must be thrown away after a single iteration of the protocol. However, by using notarized envelopes, one can greatly increase the efficiency of the protocol, as follows.

1. P uses pair-blobs to commit to an arbitrary coloring χ of G.
2. V randomly chooses an edge (i, j) of G, and sends (i, j) to P.
3. P proves to V in zero-knowledge that $\chi(i) \neq \chi(j)$.
4. To achieve a 2^{-k} error probability, P and V run Steps 2–3 $O(mk)$ times. V accepts iff he accepts in each iteration.

We now give some intuition as to why the above protocol constitutes a proof system. If G is 3-colorable, and P behaves properly, then V will always accept. If G is not three-colorable, then no matter what coloring a prover $\hat{P}$ commits to, V will choose a bad edge with probability at least $1/m$. In this case, no matter what strategy $\hat{P}$ uses, V will reject with some nonconstant probability.

We now give some intuition as to why the proof system is zero-knowledge. Regardless of which edge (i, j) of G a possibly malicious verifier $\hat{V}$ might send, his view will consist of a zero-knowledge proof that $\chi(i) \neq \chi(j)$, which can be easily simulated in the ideal envelope model. Furthermore, after each iteration of the protocol, the distribution on P's commitment to χ conditioned on $\hat{V}$'s view, is identical to the initial distribution on P's commitment to χ. Thus, after the initial commitment has been simulated, the simulation for each iteration of the protocol is essentially the same.

Finally, we note that the modified protocol is much more efficient that the original. Suppose that G has n nodes and m edges, and we wish to achieve a 2^{-k} error probability. Both protocols must be iterated $\theta(mk)$ times. Each iteration of the original requires $O(n)$ bit commitments, so $O(nmk)$ bit commitments are required in all. However, the modified protocol requires a one-time cost of $O(n)$ bit commitments, after which only $O(1)$ bits are required for each subsequent iteration. Thus, only $O(n + mk)$ bit commitments are required by our modified protocol.

3.2 Combining notarized envelopes and transparent proofs.

We were able to make the 3-colorability proof more efficient because it consisted of an expensive commitment stage followed by an inexpensive testing stage. The same is true for transparent proofs. By performing the initial commitment stage only once, we can construct zero-knowledge proofs for NP with previously unattainable resource requirements.

Let us first review the general characteristics of the transparent proofs in [4]. A verification that $x \in L$ proceeds as follows.[5]

- First, x is converted into a string $x' = \mathrm{C}(x)$, by a simple, polynomial time algorithm, and x' is given to V. This preprocessing step is assumed to be correct in the [4] scenario. V is also assumed to know $|x|$, which we denote by n.
- Prover P furnishes V with w, a transparent proof that $x \in L$.
- V generates r, a sequence of $\log^c n$ coin tosses. V then, in polylogarithmic time, computes $q = \mathrm{Q}(n, r)$, a list of polylogarithmically many indices of w and x' that he wishes to see. Thus, V can ask to see bits 5, 11 and 31 of w and bits 3 and 21 of x. Note that q is essentially independent of w and x'.
- An oracle supplies V with a sequence a, consisting of the answers to V's queries. V computes $\mathrm{A}(r, q, a)$ in polylogarithmic time, and accepts iff $\mathrm{A}(r, q, a) = 1$.

If $x \in L$, and P correctly constructs w, then V will always accept. If $x \notin L$ then V will reject with probability at least $\frac{1}{2}$, regardless of the value of w.

In Figure 1, we give a zero-knowledge proof system for L, based on the existence of a [4] transparent proof for L.

EFFICIENT-PROOF$(x, w, k, c, \mathrm{C}, \mathrm{Q}, \mathrm{A})$

1. P and V compute $x' = \mathrm{C}$, and $n = |x|$. P commits to w using pair-blobs.
2. V uniformly chooses $r \in \{0,1\}^{\lg^c n}$, and sends r to P.
3. P and V compute $q = \mathrm{Q}(n, r)$. P gives a zero-knowledge proof that $\mathrm{A}(r, q, a) = 1$, where a is defined by q, x' and the committed value of w. V accepts iff he accepts this proof.
4. To achieve a 2^{-k} error probability, P and V run Steps 2–3 $24k$ times. V accepts iff he accepts in each iteration.

Figure 1: An asymptotically efficient proof system for NP.

3.3 Properties of our protocol.

We now argue that EFFICIENT-PROOF is indeed a proof system, that it can be used to make efficient proofs for the circuit satisfiability problem (and by extension, to all of NP), and give some intuition as to why the protocol is zero-knowledge.

Our protocol is proof system.

Theorem 2 Suppose that $(c, \mathrm{C}, \mathrm{Q}, \mathrm{A})$ form a transparent proof system for L, such that

1. Correct proofs are accepted with probability 1.
2. Incorrect assertions are rejected with probability at least $\frac{1}{2}$.

Then, when EFFICIENT-PROOF$(x, w, k, c, \mathrm{C}, \mathrm{Q}, \mathrm{A})$ is run, the following properties must hold:

1. If w is a correct transparent proof for $x \in L$, then V will accept with probability 1.
2. If $x \notin L$, then V will reject with probability 2^{-k}.

Proof: If w is a correct proof for $x \in L$, then for any $r \in \{0,1\}^{\lg^c n}$, it will always be the case that $\mathrm{A}(r, q, a) = 1$, where $q = \mathrm{Q}(|x|, r)$ and a is consistent with q, x' and the committed bits of w. In this case, by Theorem 1, V will always accept P proof of this correct statement. If $x \notin L$, then for any committed w, with probability at least $\frac{1}{2}$, r will be such that $\mathrm{A}(r, q, a) \neq 1$. Whenever this case occurs, P will have to prove an incorrect assertion in Step 3 of the protocol, in which case V will reject with probability $1/12$, by Theorem 1. Thus, in each iteration of Steps 2 and 3, the probability that V will catch a false statement is at least $1/24$, regardless of any of the previous iterations. Finally, the probability that P will survive all $24k$ iterations is at most,

$$\left(1 - \frac{1}{24}\right)^{24k} \leq 2^{-k}. \quad \blacksquare$$

Our protocol is efficient.

We now show that for the circuit satisfiability problem, one can use our construction to create an asymptotically efficient proof system.

[5] Mnemonically, C, Q and A stand for CODE, QUERY and ACCEPT, whose use proved incompatible with two-column formatting.

Theorem 3 Let $\epsilon > 0$ be a fixed constant. There exists a transparent proof $(c, \text{C}, \text{Q}, \text{A})$ for the circuit satisfiability problem such that

$$\text{EFFICIENT-PROOF}(x, w, k, c, \text{C}, \text{Q}, \text{A})$$

requires $O(n^{1+\epsilon} + (\lg^{O(1/\epsilon)} n)k)$ ideal bit commitments, revelations and bits of communication.

Proof: We first bound the running time of our protocol by the various operations of the transparent proof. We then note that a construction of [4] gives us our desired result. The total number of committals, revelations and bits of communication required for the protocol is equal to that required for Step 1 of the protocol plus $24k$ times that required for Step 3 of the protocol. The number of committals required in Step 1 of our protocol is $O(|w|)$. The number of committals, revelations and communication required for the zero-knowledge proof in Step 3 of the protocol is bounded above by some polynomial in the time required to compute $\text{A}(r, q, a)$.

Since there is an $\lg^c n$ PRAM algorithm for *verifying* that an n-gate circuit is satisfiable, the general construction of [4] allows one to make a transparent proof for the circuit satisfiability problem that is of size $n^{1+\epsilon}$, where n is the size of the circuit, and $\epsilon > 0$ is some arbitrary constant. The verification time for this proof (which trivially bounds the time required to compute $\text{A}(r, q, a)$) will be $\lg^{O(1/\epsilon)} n$. Plugging these figures in, we obtain the desired resource bounds. ▌

Our protocol is zero-knowledge.

We do not have space to include a full proof that our protocol is zero-knowledge. Such a proof would require us to delve into the details of the simulator for the Brassard-Crépeau protocol as well as the simulators for the PROVE-EQUAL protocol. We give the simulator for the protocol in Figure 2. We assume that the simulator has access to $|w|$, the length of the transparent proof. The lengths of the proofs in [4] are easily constructible.

Here is some intuition for why the simulation works. In Step 1 of the protocol, the prover commits to $|w|$ bits, and in the ideal bit commitment model, $\hat{V}$'s view is the same regardless of the values of these bit. Thus, his view of Step 1 is the same as the view obtained by the simulator. In Step 2, $\hat{V}$ is just sending a string, so the equivalence of views will not be broken. The main subtlety comes in considering the effects of Step 3 of the protocol and its simulation. Before each iteration of Steps 2 and 3 of the protocol, w has some fixed value, and the pair blobs representing it are secure against $\hat{V}$ (by induction, and Part 2 of Theorem 1). Before each iteration of Steps 2 and 3 of the simulation, w will consist of all 0's, whose pair blob representations are secure against $\hat{V}$ (by induction, and the security property of the PROVE-CIRCUIT simulator).

Simulator EFFICIENT-PROOF$(x, |w|, k, c, \text{C}, \text{Q}, \text{A})$

1. S computes $x' = \text{C}$, and $n = |x|$. S commits to $0^{|w|}$ using pair-blobs.
2. $\hat{V}$ sends $r \in \{0,1\}^{\lg^c n}$ to S.
3. S uses the PROVE-CIRCUIT simulator to "fake" the zero-knowledge proof that $\text{A}(r, q, a) = 1$. (In fact, the committed bits corresponding to a are all 0's.)
4. To simulate each iteration of Steps 2–3, S runs Steps 2–3 of the simulation.

Figure 2: A simulator for the efficient proof system for NP.

Now, regardless of r, the predicate to be proven in Step 3 will be true. Since both the witness blobs and the 0 blobs (used by the protocol and simulation respectively) are secure against $\hat{V}$, the black-box simulation property of the PROVE-CIRCUIT simulator guarantees that the two views will remain equal.

As a corollary, we obtain asymptotically efficient zero-knowledge proof systems for any language in NP.

Corollary 5 For any language $L \in NP$, there exists a zero-knowledge proof system for $x \in L$ that requires only $O(n^c + \lg^c n)k)$ ideal bit commitments, revelations and bits of communication to achieve a 2^{-k} error probability. Here $n = |x|$ and c is some constant depending only on L. ▌

4 Communication efficient arguments for NP.

In this section, we show how to transform the zero-knowledge proofs in the previous section into communication efficient arguments for NP, making a reasonable complexity assumption. Whereas our proof systems require an expensive set-up cost, the arguments do not require any such cost. Indeed, even if one desires $\frac{1}{3}$ error probability, our proofs are much more communication efficient than any that were previously known. We first note that one can easily implement the previous protocols as arguments, given a secure blob system. Next, we first describe our technique for shrinking our first stage of the proof system, and then we give the resulting protocol.

4.1 A naive implementation.

Neglecting communication costs, we can straightforwardly transform the proof system in the ideal bit commitment model into an argument (or for that matter, a cryptography based interactive proof system). Let l be the security parameter for the prover. (Note that l has nothing to do with bounding the probability of error, but rather makes a tacit assumption on the computational power of the prover.) First, P and V agree on some information theoretically secure blob system, represented by the probabilistic procedures

$$\begin{aligned} \text{BLOB}_{l,k} &: \{0,1\} \rightarrow \{0,1\}^l \times \{0,1\}^l, \text{ and,} \\ \text{CHECK}_{l,k} &: \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}. \end{aligned}$$

That is, given a bit b, $\text{BLOB}_{l,k}(b)$ generates a pair (C, R), such that $\text{CHECK}(C, R) = b$. Here C is the "blob" for b and R is the string used to reveal b. P commits to b by generating (C, R) and sending C to V. P reveals b by sending R to V, who computes $\text{CHECK}(C, R)$. The zero-knowledge argument relies on the assumptions that,

1. The distribution on C induced by $\text{BLOB}_{l,k}(0)$ and $\text{BLOB}_{l,k}(1)$ are identical, and,
2. It is computationally infeasible for P to produce a triplet (C, R, R') such that $\text{CHECK}(C, R) = 0$ and $\text{CHECK}(C, R') = 1$.

The generation of the blob system may be performed once and for all, for all theorems to be proven.

To implement Step 1 of the protocol, P converts w into a pair-blob representation, and uses BLOB to commit to each of these bits. We'll denote the resulting blobs by $B_1, \ldots, B_m$. Step 2 of the protocol is performed just as before. Finally, Step 3 of the protocol is implemented as before, but again by using the secure blob system for the committals and revelations required for the proof.

4.2 The problem, and how to solve it.

As before, the real cost of the protocol lies in Step 1, which requires $O(n^{1+\epsilon} l)$ bits of communication. Steps 2 and 3 of the protocol require only $O(\log^c l)$ bits of communication, for some c. The problem is that virtually all of the bits committed to in Step 1 are not needed in a single execution of Steps 2 and 3. We could make our protocols much more efficient if we only had to pay for decommitting a bit. In the Brassard-Crépeau model, one can achieve this savings, provided that sufficiently strong cryptographic hash functions exist.

Suppose that there is family $\{F_{l,k}\}$ of polynomial time computable hash functions, such that

1. $F_{l,k}$ is a function from $\{0,1\}^{2l}$ to $\{0,1\}^l$, and
2. For every polynomial-sized circuit family $\{C_l\}$, the probability that $C_l(k)$ can produce x, y such that $F_{l,k}(x) = F_{l,k}(y)$, where $F_{l,k}$ is chosen according to some probabilistic polynomially samplable distribution, grows smaller than l^{-c} for any constant c.

PACK($l, C_1, \ldots, C_{2^n}$)

1. V agree on a suitably distributed hash function, $F_{l,k}$.
2. For $1 \leq i \leq n$ and $0 \leq j < 2^{n-i}$, define $C_j^0 = C_j$ and

$$C_j^i = F_{l,k}(C_{2j}^{i-1}, C_{2j+1}^{i-1}).$$

P sends C_0^n to V.

Figure 3: Protocol for cheaply committing to a number of bits.

In Figures 3 and 4, we show how to use such hash functions to cheaply commit to a large number of blobs, and (reasonably) cheaply reveal a single one of these blobs. The trick is to use our hash function to make a binary tree of commitments, where each leaf of the tree corresponds to one of the blobs the prover committed to. A blob corresponding to a node of a tree is a hashed representation of it two children. The prover commits to the entire tree by sending the blob corresponding to the root of the tree. Finally, the prover reveals the representation of a blob (distinct from revealing the contents of that blob) by revealing each hashed blob along the path from the root to the leaf, along with the each of these hashed blob's children.

Note that our protocol for revealing the representations of blobs will leak a great deal of extraneous information about the representations of other blobs. However, as far as the security of the proof is concerned, this leakage is irrelevant. Since there exist blob systems in the [2] that are perfectly and unconditionally secure, the prover would not mind if the verifier received copies of all the blobs, which indeed happens in the naive implementation.

Our final protocol is sketched in Figure 5.

What assumptions do we need?

In the argument we have implemented, we need to create zero-knowledge blobs and secure hash functions. Both of these requirements may be achieved with claw-free pairs of permutations [7]. That is, we need to produce function pairs (F, G) such that it is difficult to find a pair (x, y) such that $F(x) = G(y)$. For example, if $n = p_1 p_2$, where p_1, p_2 are unknown to P, and a is a random quadratic residue mod n, then one can set $F(x) = x^2 \bmod n$ and $G(y) = ay^2 \bmod n$, where

```
EXTRACT($F_{l,k}, I, C_1, \ldots, C_{2^n}$)  reveal blob $C_I$ */
  1. For $1 \le i \le n$, $P$ sends $C^{i-1}_{2\lfloor I/2^i \rfloor}, C^{i-1}_{2\lfloor I/2^i \rfloor+1}$ to $V$.
  2. For $1 \le i \le n$, $V$ checks that
         $C^i_{\lfloor I/2^i \rfloor} = F_{l,k}\left(C^{i-1}_{2\lfloor I/2^i \rfloor}, C^{i-1}_{2\lfloor I/2^i \rfloor+1}\right)$,
     and rejects if this is not the case. otherwise, $V$ recovers $C_I = C^0_I$.
```

Figure 4: Protocol for extracting a blob.

```
EFFICIENT-ARGUMENT($x, w, l, c$, C, Q, A)
  0. $P$ and $V$ agree on an information theoretically secure blob committal scheme, BLOB and
     CHECK, and a cryptographically secure hash function, $F_{l,k}$.
  1. $P$ and $V$ compute $x' = \mathrm{C}(x)$, and $n = |x|$. $P$ breaks $w$ into using pair-blobs,
     and uses BLOB to generate blobs $(C_0, R_0) \ldots, (C_m, R_m)$, and commits to
     $C_0, \ldots, C_m$ using protocol PACK. Here, $m$ is assumed to be a perfect power of 2.
  2. $V$ uniformly chooses $r \in \{0,1\}^{\lg^c n}$, and sends $r$ to $P$.
  3. $P$ and $V$ compute $q = \mathrm{Q}(n, r)$. $P$ gives a zero-knowledge proof that
     $\mathrm{A}(r, q, a) = 1$, where $a$ is defined by $q, x'$ and the committed value of $w$.
     Whenever $P$ has to reveal a bit represented by a blob $C_I$, he first runs
         EXTRACT($F_{l,k}, I, C_1, \ldots, C_m$),
     and then sends $R_I$ to $V$. $V$ recovers the revealed bit by computing CHECK($C_I, R_I$).
     $V$ accepts iff he accepts this proof.
```

Figure 5: An efficient argument for NP.

$x, y \neq 0$ [7]. If one could find a colliding pair, (x, y), then one can find a square root of a, namely xy^{-1}. Of course, the verifier must convince P that a is a quadratic residue; otherwise the proof will no longer be zero-knowledge.

Somewhat surprisingly, the communication complexity of setting up the blob scheme and the hash function may well dominate the communication complexity of the protocol. This is not because it is any harder to set up such schemes in our framework, rather every other step of our protocol now has a very low communication complexity. For example, the cost of proving to P that a is a quadratic residue mod n is quite large, though it need be done only once for all time (that is, for all theorems to be proven).

A really fast way of setting up a claw-free pair of permutations was proposed by Boyar, Krentel and Kurtz [5], and independently by Damgård. Suppose that it is hard to compute discrete logarithms modulo a random prime p, even given the factorization of $p-1$. Then given g, a generator for Z_p^*, and a chosen at random from Z_p^*, one can set $F(x) = g^x$ and $G(y) = ag^y$. If $F(x) = G(y)$, then $\lg^g y = x - y$. V can choose p, g, a such that $p-1$ is of known factorization, and then send this information to P. Given the factorization of $p-1$, P can trivially verify that g is a generator for Z_p^*. The entire protocol therefore takes only $O(l)$ bits of communication.

The communication complexity of our proof system.

Using the most efficiently constructible schemes we know (such as the one mentioned above, based on a strong discrete-log assumption), Steps 0 and 1 require only $O(l)$ bits of communication. As before, Step 2 requires only $\lg^c n$ bits of communication. Step 3 requires $\log^{c_1} n$ committal and revelations, for some c_1. Now each such operation requires at most $O(\lg(n)l)$ bits of communication, so the total amount of communication required for the protocol is $O(\lg^c(n)l)$, for some constant c.

5 Acknowledgments.

I would like to thank Mihir Bellare and Steven Rudich for valuable comments and insights.

References

[1] J. Boyar, G. Brassard and R. Peralta. Subquadratic Zero-knowledge Proc. of FOCS91

[2] G. Brassard and C. Crépeau. Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond, Proc. of FOCS86.

[3] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge, *J. Comput. System Sci.* **37** (1988), 156–189.

[4] L. Babai, L. Fortnow, L. Levin and M. Szegedy. Checking computation in polylogarithmic time. Proc. of STOC91

[5] J. Boyar, M. Krentel and S. Kurtz. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology,* Vol. 2, No. 2, pp. 63–76, 1990.

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

[7] S. Goldwasser, S. Micali, and R. Rivest. A Paradoxical Solution to the Signature Problem. Proc. of. FOCS84

[8] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design, Proc. of FOCS86.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game, Proc. of STOC87.

[10] R. Impagliazzo and M. Yung. Direct Minimum-Knowledge Computation, Proc. of CRYPTO87.

[11] J. Kilian, S. Micali and R. Ostrovsky. Minimum Resource Zero-Knowledge Proofs, Proc. of FOCS89.