J. Cryptology (2000) 13: 449–472
DOI: 10.1007/s001450010011

Journal of
CRYPTOLOGY
© 2000 International Association for Cryptologic Research

Short Non-Interactive Cryptographic Proofs[*]

Joan Boyar
Department of Mathematics and Computer Science,
University of Southern Denmark,
Odense, Denmark
joan@imada.sdu.dk

Ivan Damgård
Department of Computer Science, BRICS,[†] Aarhus University,
DC-8000 Aarhus C, Denmark
ivan@daimi.au.dk

René Peralta
Department of Computer Science, Yale University,
New Haven, CT 06520-8285, U.S.A.
peralta-rene@cs.yale.edu

Communicated by Joan Feigenbaum

Received 30 March 1998 and revised 29 November 1999
Online publication 10 August 2000

Abstract. We show how to produce short proofs of theorems such that a distrusting Verifier can be convinced that the theorem is true yet obtains no information about the proof itself. We assume the theorem is represented by a boolean circuit, of size m gates, which is satisfiable if and only if the theorem holds. We use bit commitments of size k and bound the probability of false proofs going undetected by 2^{-r}. We obtain non-interactive zero-knowledge proofs of size $O(mk(\log m + r))$ bits. In the random oracle model, we obtain non-interactive proofs of size $O(m(\log m + r) + rk)$ bits. By simulating a random oracle, we obtain non-interactive proofs which are short enough to be used in practice. We call the latter proofs "discreet."

Key words. Cryptographic proofs, Non-interactive proofs, Discreet proofs, Circuit complexity, Multiplicative complexity.

[*] Some of this work was carried out while Joan Boyar was visiting the Computer Sciences Department at the University of Wisconsin–Madison. It was supported in part by SNF (Denmark) and in part by NSF (U.S.) Grant CCR-9510244. The work by René Peralta was supported in part by NSF Grants CCR-9207204, CCR-9712109, and by PFU Endowed Chair at Japan Advanced Institute of Science and Technology.

[†] Basic Research in Computer Science, center of the Danish National Research Foundation.

1. Introduction

The main goal of this work is to produce short proofs of theorems such that a distrusting Verifier can be convinced that the theorem is true yet obtains no information about the proof itself. Rather than looking at the variety of proofs one could use for specific theorems, we consider proofs of circuit satisfiability, since the proof of any theorem can be transformed into a proof that some specific circuit is satisfiable. Of course any NP-Complete problem could be chosen; Goldreich et al. [21] showed that, under certain cryptographic assumptions, all problems in NP have zero-knowledge proofs. For the sake of efficiency, we choose circuit satisfiability because most problems used for cryptographic purposes are arithmetic, number theoretic, or substitution-permutation ciphers, all of which have natural circuit implementations.

In recent years there has been considerable progress in techniques for reducing the communication complexity (number of bits sent) necessary for interactive zero-knowledge proofs of circuit satisfiability [8], [6], [13], [14], [16], [20], [25]. In this paper we focus on minimizing the size of non-interactive proofs. Some of our basic ideas have appeared in preliminary form at the 1996 EuroCrypt conference [9]. The results we obtain here compare favorably with those in [15] and also with other attempts, such as [26].

In what follows, we assume that the Prover and Verifier have a circuit containing m gates with fan-in at most two. The Prover wants to convince the Verifier that the circuit is satisfiable, and the Verifier will accept a probability of at most 2^{-r} of the Prover successfully "proving" satisfiability in cases where the circuit is not actually satisfiable. The Prover does not want the Verifier to learn anything about the satisfying assignment. In order to prevent this, she[1] hides bits using a bit commitment scheme. The bit commitment scheme should satisfy standard cryptographic properties (see [11], [7], and [8]). Further properties are required and are described in the appropriate sections. In all cases, though, each bit commitment is assumed to have length k. Since the bit commitments used in this paper are such that the length of a bit commitment determines the security of the scheme, r and k are both security parameters: r determining the probability that the Prover can convince the Verifier of something false, and k determining the probability that the Verifier can learn something from the proof.

We present three related types of proofs for circuit satisfiability. The first type is non-interactive zero-knowledge in the shared random string model proposed by Blum et al. [4]. The second, which is significantly more efficient, is zero-knowledge in the random oracle model.[2] The final type is obtained by applying the random oracle heuristic (see [1]) to the second type of proofs. We call the latter type *discreet proofs*, to distinguish them from zero-knowledge proofs, while indicating that they are secure given an appropriate implementation.

[1] It has become common usage to name the Prover "Peggy" and the Verifier "Vic." Hence, "she" is the Prover and "he" is the Verifier.

[2] Both shared random strings and random oracles are trusted sources of random bits. The difference is that with shared random strings, all of the bits are available at step 1 of the protocol, while with random oracles, the bits for step $i + 1$ are not known at step i. A random oracle has no finite length, while a shared random string is polynomial in length and thus can be agreed upon in advance.

As a concrete example of a bit commitment scheme which has the properties we need, we use the quadratic residuosity (QR) bit commitment scheme [12] which is based on the Quadratic Residuosity Assumption (QRA) [22]. Throughout the paper we assume that the Prover and Verifier have already established the main parameter for the QR-bit commitment scheme (a Blum integer N, see Definition 1); we do not consider the cost of setting up this parameter. The length of our non-interactive zero-knowledge proof is $O(km(\log m + r))$ for a circuit of size m, and bounding the probability of undetected cheating by 2^{-r}. This asymptotic complexity was also obtained in [15], assuming the QRA, and impressively by Kilian and Petrank [26], who base their results on much more general assumptions. The results in both [15] and [26] are also based on circuit complexity. The advantage of our non-interactive proofs is in the constant factors, which are of course of practical interest. Our system beats the one in [15] by a factor of about 7.5, and Petrank's and Kilian's by a very large factor.

In the random oracle model, we are able to construct non-interactive zero-knowledge proofs of size $O(m(\log m + r) + rk)$ bits, again with an error probability bounded above by 2^{-r}. Proofs of the same size are then obtained by simulating the randomness source. The resulting proofs are efficient, non-interactive, and secure under standard cryptographic assumptions (as in [1]), but no longer zero-knowledge. We call these proofs "discreet." As an example of the power of these techniques, we show discreet proofs of knowledge of RSA and DES which are short enough for practical applications (e.g., computer password schemes which are immune to eavesdroppers).

The basis for these results is what we call "certificates," which are non-interactive proofs that a vector of bit commitments encodes a vector of bits in a given set without revealing the bit vector itself. We have high hopes for what useful results this technique might eventually yield.

Before continuing, we provide a notation index for the reader's easy reference.

N: the Blum integer created by the Prover.
n: security parameter; the probability that the Prover can cheat on the inputs and output of any single gate must be $O(2^{-n})$.
k: the length of bit commitments. When using QR-bit commitments, k is the length of the modulus N. In this case, $k = 1024$ is considered secure.
m: number of gates in the circuit.
r: security parameter; the probability that the Prover can cheat on the inputs and output of at least one gate must be $O(2^{-r})$.
μ: a random quadratic non-residue modulo N, chosen by the Prover.
r': security parameter; the probability that the Prover can falsely convince the Verifier that μ is a quadratic non-residue must be $O(2^{-r'})$.
$\mathcal{D}(\vec{x})$: the vector in GF_2^n encoded by the vector of commitments $\vec{x}$.
$\hat{\vec{x}}$: short notation for $\mathcal{D}(\vec{x})$.
$\mathcal{C}(\vec{s})$: a set of commitments to the bit vector $\vec{s}$.
$c_\wedge(f)$: the conjunctive complexity of the function f.
$COMM\_0$: an array of bit commitments. The Prover claims they are all commitments to 0.
Z_n^+ (Z_n^-): the set of quadratic residues (non-residues, resp.) modulo N.

2. The Model

We use the standard model of Goldwasser et al. [23] consisting of interactive Turing machines (one called the “Prover” and the other the “Verifier”). This model can have several instantiations, depending on the power these machines have. In this paper the Verifier is constrained to probabilistic polynomial-time computation. The Prover can have unlimited computing power or be constrained to probabilistic polynomial-time computation. In the latter case, the Prover has access to an auxiliary tape which is not available to the Verifier. This tape will typically contain a satisfying assignment to the circuit that the Prover claims is satisfiable.

Generally, establishing a bit commitment scheme is the first part of a (non-interactive) proof. In our case, this essentially means that the Prover should generate a Blum integer N (see Definition 1) and convince the Verifier that N is indeed a Blum integer. For simplicity, and to focus on the design of the actual proofs, we assume throughout that a suitable N is already available, and do not consider the cost of generating and validating it.[3] Our techniques do not, however, preclude doing this at the beginning of every proof. Doing so would simply make the proofs longer.

The assumption that N is available is reasonable in an environment where cryptographic keys are widely used, and it allows us to assume that the shared random string is independent of N. Note that this implies that the shared random string must be generated after N has been published by the Prover. This means previous suggestions, such as using the binary expansion of π, or using available tables of random bits, will not work (unless we are willing to generate a longer non-interactive proof which is secure even if N is generated and validated at the same time as the theorem is being proved, see [15]). It should also be noted here that our “non-interactive zero-knowledge proofs” are bounded in the sense of [5], i.e., the Prover cannot prove more than one SAT instance using the same shared string. In [5] and [18], methods are given for transforming a protocol such as ours into one in which the Prover can prove arbitrarily many instances of any size. It is not difficult to apply these techniques to our protocols, but they do, of course, increase the length of the proofs. If we replace the shared random string model by the random oracle model, we can produce much shorter proofs.

Finally, we note that in practice, and at the cost of losing the zero-knowledge property, we can either simulate the random oracle or use a trusted third party as a source of random bits. Simulation of the random oracle is used in Section 9 to obtain *very short* non-interactive discreet proofs.

3. The Bit Commitment Scheme

A bit commitment scheme is a protocol by which one party can encrypt a bit b as a string t (called the bit commitment) so that another party cannot tell whether t is the encryption of a zero or a one, but the first party is committed in the sense that she can show that t is the encryption of b but cannot show that it is an encryption of the

[3] An efficient interactive protocol for showing that N is a Blum integer can be found in [33]. A non-interactive zero-knowledge proof that N is a Blum integer can be found in [27].

complement $1 - b$. The first such scheme was suggested by Blum [2]. We assume that a bit commitment scheme is available, and that the scheme is unconditionally binding and computationally concealing. In addition to the standard cryptographic properties (see [11], [7], and [8]), we assume the bit commitment scheme allows *non-interactive processing* of XOR gates. This means that, given two bit commitments t_1 and t_2, the two parties can both (independently) compute the same bit commitment t_3 which is guaranteed to encrypt the sum modulo 2 of the bits encrypted by t_1 and t_2 (see [8] and [6]). We also assume the "trapdoor" property, which means that the Prover (committer) can interpret any bit string of the appropriate length as a bit commitment and open it, no matter how that string was produced.

As a concrete example, we choose the well-known bit commitment scheme based on the QRA. The QRA was originally stated in Goldwasser and Micali in [22] for use in probabilistic encryption. Goldreich et al. were the first to use this probabilistic encryption for bit commitments in zero-knowledge proofs [21]; Brassard and Crépeau [12] were the first to exploit the properties of this bit commitment scheme which make the non-interactive processing of XOR gates possible; and De Santis et al. in [28] were the first to use QR-bit commitments in non-interactive zero-knowledge proofs using the trapdoor property.

Definition 1. A Blum integer is a composite integer $N = P^r Q^s$, with P and Q primes such that $P \equiv Q \equiv 3 (\text{mod } 4)$ and r and s are odd.

Definition 2. Let N, μ be integers produced by the Prover, such that N is a Blum integer and μ is a non-square modulo N with Jacobi symbol 1. QR-bit commitments are constructed as follows: the Prover selects a random $x \in Z_n^*$. To commit to a 0, the Prover produces $x^2 \bmod N$. To commit to a 1, the Prover generates $\mu x^2 \bmod N$.

Definition 3. A trapdoor QR-bit commitment scheme is a QR-bit commitment scheme based on a Blum integer $N = P^r Q^s$, where the Prover knows the secrets P and Q, together with a public integer β with Jacobi symbol -1 modulo N. Then $x \in Z_n^*$ is the commitment to a 0 if x or $\beta x \bmod N$ is a quadratic residue modulo N; otherwise it is a commitment to a 1.

The advantage of a trapdoor QR-bit commitment scheme (over other QR-bit commitment schemes) is that with such a scheme, the Prover can interpret any integer z, $1 \leq z \leq N - 1$, which is relatively prime to N as a bit commitment. Furthermore, the Prover can open that commitment, even though she may not have created it herself.

Assuming μ is a non-square, this scheme is unconditionally binding; it is also concealing under the QRA, which states that random squares and non-squares of Jacobi symbol $+1$ modulo a Blum integer are computationally indistinguishable. We can now easily verify that the QR-bit commitment scheme allows non-interactive processing of XOR gates: if a and b are bit-commitments, then $ab \bmod N$ is a bit commitment to $\hat{a} \oplus \hat{b}$.

When using the commitment scheme in our protocols, we assume as mentioned that a suitable N is available. Known techniques then allow for proving that the parameter μ is a non-square modulo N as required, and for the extraction of secure trapdoor QR-bit commitments from a shared random string. For the sake of completeness we show, in Section 3.1, ways of doing this.

Notation. If $\vec{x} = (x_1, \ldots, x_n)$ is a vector of bit commitments, then $\mathcal{D}(\vec{x})$ denotes the vector in GF_2^n encoded by $\vec{x}$. If $\vec{s} \in GF_2^n$, we denote by $\mathcal{C}(\vec{s})$ the set of commitments to $\vec{s}$. To indicate that $\vec{x}$ is chosen uniformly at random from the set $\mathcal{C}(\vec{s})$, we write $\vec{x} \in_R \mathcal{C}(\vec{s})$. For readability, we also employ the notation $\vec{\hat{x}}$ to denote $\mathcal{D}(\vec{x})$. Denote by Z_n^+ (resp. Z_n^-) the subset of Z_n composed of all elements of Jacobi symbol 1 (resp. -1).

3.1. *Obtaining QR-Bit Commitments and the Parameter μ from Shared Random Strings*

Given a Blum integer N for which the Prover knows the factors P and Q, we would like to establish a trapdoor QR-bit commitment scheme using a shared random string.

To do this, one first chooses random values modulo N until one is found with Jacobi symbol -1. Then that can be published as β. The Prover will similarly choose μ, a random non-square in Z_n^+.

Then, to validate μ, and to extract commitments from the shared random string, we need to be able to extract numbers of Jacobi symbol 1. To do this, one interprets each k-bit block in the random string as a number modulo N. Numbers larger than N are discarded. Those which are not discarded and are in Z_n^+ can be directly used. Those which are not discarded and are in Z_n^- can be used after multiplication by β. On average, no more than half of the k-bit blocks need to be discarded.

Since QR-bit commitments are always numbers of Jacobi symbol 1, this procedure can be used to generate bit commitments, once the parameter μ has been validated. We now show, using a slight modification of a method from [5], how the Prover can give a non-interactive proof that μ is a non-square. The Prover will have a probability of at most $2^{-r'}$ of convincing the Verifier that a square is a non-square.

Prover's Algorithm

1. The Prover extracts $r' + 1$ numbers of Jacobi symbol 1 modulo N from the shared random string. She splits the numbers into two sets A, B such that a number is in A if and only if it is a square modulo N. She sets $W = A$ or B, choosing between the two possibilities at random, and indicates the set W in her proof.
2. Let a_0 be the first number in A, b_0 the first in B. The Prover includes in the proof a square root modulo N of $a_0 a_i \bmod N$ for every $a_i \neq a_0$ in A, and similarly a root of $b_0 b_j \bmod N$ for every $b_j \neq b_0$ in B. Finally, she includes a square root of $\mu a_0 b_0 \bmod N$.

Verifier's Algorithm

1. The Verifier extracts r' numbers of Jacobi symbol 1 modulo N from the shared random string. He splits the numbers into two sets, namely, the W indicated in the non-interactive proof, and its complement.
2. The Verifier checks every square root provided by the Prover against the appropriate products of the numbers just extracted. The proof is accepted if and only if all square roots are correct, and both W and its complement are non-empty.

Note that, except with probability $2^{-r'}$, there will be both squares and non-squares in the extracted set. In this case the honest Prover can convince the Verifier because both A and B will be non-empty and all the numbers a_0a_i, b_0b_j, $\mu a_0 b_0 \bmod N$ will be squares.[4] Thus, in this case it is impossible to have a square accepted as μ, since acceptance of a square immediately implies that all extracted numbers have the same quadratic character. The length of the proof is easily seen to be at most $r'k + r' + 1$ bits.

We do not prove this procedure to be zero-knowledge here. It will be used as a subroutine in our main protocol for satisfiability, which is shown to be zero-knowledge in its entirety in Section 6.

The construction we described for extracting numbers from the shared string is straightforward and will be the one we assume is used when we prove the zero-knowledge property. In practice, the random string can be used more efficiently by making two simple observations. First, if we make the modulus N congruent to 5 modulo 8, then we can set $\beta = 2$. Second, the QR problem is self-randomizable. That is, a number $x \bmod N$ is a quadratic residue if and only if $xy^2 \bmod N$ is a quadratic residue for all y. This implies that if the QRA holds, then it holds over any (sufficiently large) subset of the elements of Z_n^+. Thus, we may assume the QRA holds over $\{x \mid x \in Z_n^+ \text{ is of length } k-1 \text{ bits}\} \cup \{2x \mid x \in Z_n^- \text{ is of length } k-1 \text{ bits}\}$. Thus we may safely extract one bit commitment for each block of $k-1$ bits of the shared random string.[5]

4. Constructible and Certifiable Sets

In this section we show how the Prover can create bit commitments corresponding to the inputs and outputs of an arbitrary binary gate and convince the Verifier that the bits committed to correspond to some row in the truth table for that gate. This will be the key to our non-interactive zero-knowledge proofs and our discreet proofs of circuit satisfiability. Consider for example an AND gate. The four rows of the truth table for AND give rise to the set $S_\wedge = \{000, 010, 100, 111\}$. The inputs and output of any AND gate must correspond to one of these four elements. In our non-interactive zero-knowledge proofs of circuit satisfiability, for each AND gate, the Prover will commit to the appropriate member of this set and give a (non-interactive) proof showing that the commitments are to some member of $S_\wedge$. We call such proofs *certificates*. Commitments and certificates for different members of $S_\wedge$ will be computationally indistinguishable from each other, so the Verifier will learn nothing about which element of $S_\wedge$ is used.

Constructing commitments and certificates for members of the set $S_\wedge$ is nontrivial. We build towards this goal by first showing how to construct commitments and certificates for the two "easier" sets $T = \{01, 10, 11\}$ and $U = \{0111, 1011, 1101, 1110\}$. First, we formalize the notions of constructibility and certifiability of sets.

[4] In our application of this, the Prover chooses μ herself. If one wants a protocol where the honest Prover can *always* convince the Verifier, one can adopt the convention that the Prover chooses μ as a square in case all extracted numbers have the same quadractic character, and then continues as usual.

[5] The reader may have noticed that the probability distribution of bit commitments extracted in this way may vary slightly from the uniform distribution. This is not a problem, again because of the self-randomizable property of the QR problem.

Throughout this section we assume that the bit commitment scheme has been established, and, for simplicity, we assume that the probability of incorrect parameters for the scheme is zero. In the case of the trapdoor QR-scheme, we assume that the Blum integer N and the non-square μ are correctly chosen, meaning with zero probability that N is not a Blum integer and with zero probability that μ is a square. We also assume that the shared random string was not used in the creation or verification of these parameters. This is just a simplification. In a practical scenario, N and μ would be chosen by the Prover and shown to be correctly chosen by a zero-knowledge proof. The actual error probability of the overall protocol is then at most the sum of the probabilities of all the error cases.

Our non-interactive zero-knowledge proofs require shared randomness, but we first consider what can be done without it.

Definition 4. A subset S of GF_2^n is said to be constructible if it is possible for the Prover to construct, for any chosen $\vec{s} \in S$, a vector $\vec{x}$ of n bit commitments, along with a certificate c showing that $\mathcal{D}(\vec{x})$ belongs to S. It must be the case that:

- If the Prover follows its algorithm, $\mathcal{D}(\vec{x}) = \vec{s}$.
- If the Prover and Verifier both follow their algorithms, the Verifier will accept c.
- If a Verifier following its algorithm accepts c, then $\mathcal{D}(\vec{x}) \in S$.
- Certificates for the different elements of S must be computationally indistinguishable.

Notice that no interaction is used, and the commitments and certificate are constructed without using the shared random string. Since certificates for different elements of S are computationally indistinguishable, the certificate does not reveal any information about which element of S is $\vec{s}$.

For example, the existence of bit commitment schemes implies that the set $S = \{0, 1\}$ is trivially constructible since no certificate is needed (more formally, the empty string is a certificate). If the bit commitment scheme allows non-interactive processing of XOR gates, then the sets $E = \{00, 11\}$ and $D = \{01, 10\}$ are also constructible. In the case of trapdoor QR-bit commitments, given commitments $\vec{x} = (x_1, x_2)$, a certificate for $\mathcal{D}(\vec{x}) \in E$ is simply a square root of $x_1 x_2 \bmod N$. A certificate for $\mathcal{D}(\vec{x}) \in D$ is a square root of $\mu x_1 x_2 \bmod N$. For details and security proofs related to this scheme see [8].

Definition 5. A subset S of GF_2^n is said to be certifiable if, given any vector $\vec{x}$ of n bit commitments such that $\mathcal{D}(\vec{x}) \in S$, the Prover can construct a certificate c showing that $\mathcal{D}(\vec{x})$ belongs to S. It must be the case that:

- If the Prover and Verifier both follow their algorithms and $\mathcal{D}(\vec{x}) \in S$, the Verifier will accept c.
- If a Verifier following its algorithm accepts c, then $\mathcal{D}(\vec{x}) \in S$.
- Certificates for the different elements of S must be computationally indistinguishable.

The difference between certifiable and constructible sets is that, with a certifiable set, the Prover has not created the bit commitments herself. This gives her less freedom in producing the proof. Note that a set which is certifiable is also constructible. Also note that under the QRA, the sets E and D are certifiable using trapdoor QR-bit commitments. In fact, for any bit commitment scheme in which E is certifiable, any set which is constructible is also certifiable. To see this, suppose $\vec{x} = (x_1, \ldots, x_n)$ is given, and that the Prover wants to certify that $\mathcal{D}(\vec{x})$ belongs to a constructible set S. To do this, the Prover simply constructs $\vec{y} = (y_1, \ldots, y_n)$ such that $\mathcal{D}(\vec{x}) = \mathcal{D}(\vec{y})$ and a certificate c for $\vec{y}$. In addition, the Prover certifies that $\mathcal{D}(x_i)\mathcal{D}(y_i) \in E$ for each i. Thus, *under the QRA, certifiable sets and constructible sets are the same*.

Lemma 1. *Under the QRA, any subspace S of GF_2^n is certifiable using trapdoor QR-bit commitments.*

Proof. Since S is a subspace of GF_2^n, it must be the image of GF_2^n under multiplication by an $n \times n$ matrix M over GF_2. To certify a vector $\vec{x}$ of n bit commitments, the Prover chooses $\vec{y}$ such that $M\vec{y} = \mathcal{D}(\vec{x})$ and constructs bit commitments $\vec{z} \in_R \mathcal{C}(\vec{y})$. Then the Verifier and Prover can non-interactively compute bit commitments $\vec{v}$ such that $\mathcal{D}(\vec{v}) = M(\mathcal{D}(\vec{z}))$ (see [6] for more details). The Prover then gives certificates showing that $\mathcal{D}(v_i)\mathcal{D}(x_i) \in E$ for each i. □

By a *linear map* from GF_2^m to GF_2^n, we mean a function $f(x) = M\vec{x} + \vec{s}$, where M is an $n \times m$ matrix over GF_2 and $\vec{s} \in GF_2^n$. Sets of the form $f(S)$ where S is a subspace of GF_2^m are called *cosets* (each such set is a coset of the subspace $\{\vec{v} \mid \vec{v} = M\vec{x}; \vec{x} \in S\}$). The proof of the following lemma is analogous to the proof of Lemma 1. The details are left to the reader.

Lemma 2. *Under the QRA, cosets are certifiable using trapdoor QR-bit commitments.*

If, for example, the set $S_{\overline{\wedge}} = \{001, 011, 101, 110\}$, which contains the rows of a truth table for a NAND gate, was certifiable, it would follow that a non-interactive zero-knowledge proof for circuit satisfiability is possible without using the shared random string at all, except to set up the bit commitment scheme. The Prover would convert its original circuit to one with just NAND gates, calculate from a satisfying set of inputs the values on every wire, create random commitments to all of those values, certify that the inputs and output of every gate correspond to a row in the truth table for a NAND gate, and finally open the bit commitment on the output wire to show that it encrypts a one. Unfortunately we have not been able to construct certificates for $S_{\overline{\wedge}}$. It may well be that if we restrict ourselves to QR-bit commitments, then cosets are the only certifiable sets. Other bit commitment schemes can be constructed for which sets corresponding to truth tables of various boolean functions are certifiable. However, we have not been able to construct a bit commitment scheme such that a logically complete set of boolean functions is certifiable.

The next best thing to constructible/certifiable sets are sets which can be constructed/certified using the shared random string. This string provides a source of randomness which the Verifier trusts, even though there is still no interaction. This shared randomness introduces a small probability that the Prover can cheat in the actual proof, in addition possibly to cheating when setting up the bit commitment scheme. Thus there is some chance that the Verifier will accept a certificate even though the vector committed to does not belong to the set S, but that probability will be negligible.

Definition 6. A subset S of GF_2^n is said to be shared-randomness constructible if it is possible for the Prover, for any chosen $\vec{s} \in S$, to construct, from an initial segment $\vec{I}$ of the shared random string, a vector $\vec{x} \in_R \mathcal{C}(\vec{s})$ of n bit commitments, along with a certificate c showing that $\mathcal{D}(\vec{x})$ belongs to S. It must be the case that:

- If the Prover follows its algorithm, $\mathcal{D}(\vec{x}) = \vec{s}$.
- If the Prover and Verifier both follow their algorithms, the Verifier will accept c.
- If $\mathcal{D}(\vec{x}) \notin S$, a Verifier following its algorithm rejects c with overwhelming probability.
- Triples $(\vec{I}, \vec{x}, c)$ for different elements of S are computationally indistinguishable.

Definition 7. A subset S of GF_2^n is said to be shared-randomness certifiable if, given any vector $\vec{x}$ of n bit commitments such that $\mathcal{D}(\vec{x}) \in S$, the Prover can construct a certificate c, from an initial segment $\vec{I}$ of the shared random string, showing that $\mathcal{D}(\vec{x})$ belongs to S. It must be the case that:

- If the Prover and Verifier both follow their algorithms, the Verifier will accept c.
- If $\mathcal{D}(\vec{x}) \notin S$, a Verifier following its algorithm rejects c with overwhelming probability.
- Triples $(\vec{I}, \vec{x}, c)$ for different elements of S are computationally indistinguishable.

We will show that under the QRA, the set $S_{\overline{\wedge}}$ is shared-randomness constructible using trapdoor QR-bit commitments. This will be used in designing non-interactive zero-knowledge proofs for circuit satisfiability, using a shared random string and in designing the more efficient discreet proofs. However, in order to construct a member of $S_{\overline{\wedge}}$, we first construct the corresponding member of the set $U = \{0111, 1011, 1101, 1110\}$, and in order to construct that member of U, we construct two members of the set $T = \{01, 10, 11\}$. We now consider these sets T and U and show that they are shared-randomness constructible under the QRA.

4.1. *The Set* $T = \{01, 10, 11\}$

To construct the set T, the Prover, when given $\vec{t} \in T$, will use the shared random string to construct a vector $\vec{b} = (b_1, b_2)$ of trapdoor QR-bit commitments such that $\vec{t} = \mathcal{D}(\vec{b})$, along with a certificate that $\vec{t}$ belongs to T. The Prover and Verifier use a parameter n which will serve to bound the probability that the Prover can issue a false certificate.

Prover's Algorithm

0. Use the shared random string to determine a sequence $\vec{x} = (x_1, \ldots, x_n)$ of n bit commitments.
   (Recall that $\vec{\hat{x}}$ is used to denote $\mathcal{D}(\vec{x})$.)
1. Choose a random n-bit vector $\vec{\hat{y}} \notin \{\vec{0}, \vec{\hat{x}}\}$.

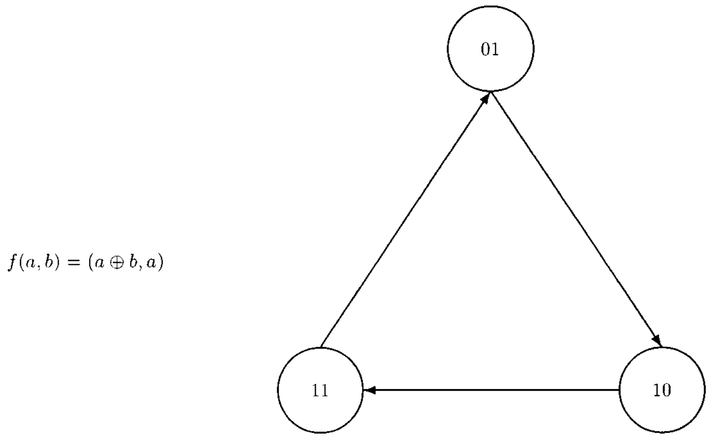

Fig. 1. Going from random commitments to chosen commitments in the set T.

2. Compute $\vec{\hat{z}} = \vec{\hat{x}} \oplus \vec{\hat{y}}$. Find i, j such that $\{\hat{x}_i\hat{x}_j, \hat{y}_i\hat{y}_j, \hat{z}_i\hat{z}_j\} = \{01, 10, 11\}$, using the method described below. If such a pair does not exist, go back to Step 1 and choose another $\vec{\hat{y}}$.
3. Choose, uniformly at random, two non-zero vectors $\vec{\hat{u}}$, $\vec{\hat{v}}$ from the set $A = \{\vec{0}, \vec{\hat{x}}, \vec{\hat{y}}, \vec{\hat{z}}\}$. Set $\vec{\hat{u}}$, $\vec{\hat{v}}$ at the beginning of the certificate.
4. Compute an $(n-2) \times n$ matrix M such that the set A is the kernel of the linear homomorphism $h(\vec{w}) = M\vec{w}$, i.e., $\vec{w} \in A$ if and only if $M\vec{w} = \vec{0}$.
5. Compute $\vec{\Psi} = (\Psi_1, \ldots, \Psi_{n-2})$, where $\Psi_i = (\prod_{\{j \mid m_{i,j}=1\}} x_j) \bmod N$. Note that the Ψ_i's are bit-commitments such that $\vec{\hat{\Psi}} = M\vec{\hat{x}} = \vec{0}$.
6. Open the $n-2$ bit commitments in $\vec{\Psi}$, and append the resulting $n-2$ numbers modulo N to the certificate.
7. The two bit commitments x_i and x_j are now commitments to a *random* member of T. They can be transformed into bit commitments for the *given* $\vec{t} \in T$ using the transformation $f(a, b) = (a \oplus b, a)$. Figure 1 shows the effect of this linear transformation. The operation $\oplus$ on bits is implemented by multiplying the respective bit commitments modulo N. The Prover only needs to append two bits to the certificate, indicating how many times to apply the linear transformation f.

In Step 2, the Prover uses the following method to choose i and j such that $\{\hat{x}_i\hat{x}_j, \hat{y}_i\hat{y}_j, \hat{z}_i\hat{z}_j\} = \{01, 10, 11\}$.

- Let $i = \min\{i \mid x_i = 1 \vee y_i = 1\}$.
- If $\hat{x}_i = 1$ and $\hat{y}_i = 0$, let $j = \min\{j > i \mid \hat{y}_j = 1\}$.
- If $\hat{x}_i = 0$ and $\hat{y}_i = 1$, let $j = \min\{j > i \mid \hat{x}_j = 1\}$.
- If $\hat{x}_i = 1$ and $\hat{y}_i = 1$, let $j = \min\{j > i \mid \hat{x}_j \neq \hat{y}_j\}$.

Note that the same i and j are calculated regardless of which two of the three possibilities for $\hat{u}$ and $\hat{v}$ get chosen. Since $\hat{x}$ and $\hat{y}$ are both random vectors and the only restriction on $\hat{x}_i$ and $\hat{y}_i$ is that they are not both zero, the other three possibilities are equally likely. Given any one of those three possibilities, there are two possibilities for the pair $\hat{x}_j, \hat{y}_j$ and they are equally likely. Thus, the quadruple $\hat{x}_i, \hat{y}_i, \hat{x}_j, \hat{y}_j$ will be one of the following {0110, 0111, 1001, 1011, 1101, 1110} values, all of which are equally likely. Thus, the pair $\hat{x}_i, \hat{x}_j$ will be an element of T, and it is equally likely to be any of the three possibilities.

Note that the above description of the Prover's algorithm assumes that $\vec{\hat{x}}$ is not the zero vector. If it is the zero vector, $\vec{\hat{u}}$ and $\vec{\hat{v}}$ can be any vectors which generate a subspace A such that the i and j from Step 2 can be found, and that subspace A can be used in what follows. Thus an honest Prover could continue with the proof. On the other hand, this unlikely situation would allow a dishonest Prover to cheat and commit to the string 00.

Verifier's Algorithm

0. Use the shared random string to determine exactly the same vector $\vec{x} = (x_1, \ldots, x_n)$ as the Prover.
1. Use the vectors $\vec{\hat{u}}$ and $\vec{\hat{v}}$ to compute the same i, j, M, and $\vec{\Psi}$ as the Prover.
2. Check that $\vec{\Psi}$ is correctly opened as the zero vector. Reject if this is not the case.
3. Determine the bit commitments $\vec{t}$ by applying f the number of times indicated by the last two bits of the certificate.

By choosing the two vectors $\vec{\hat{u}}$ and $\vec{\hat{v}}$ randomly from among the three possibilities, the Prover gives away no information about which of the vectors in A is $\vec{\hat{x}}$, but A is still completely specified since it must equal $\{\vec{0}, \vec{\hat{u}}, \vec{\hat{v}}, \vec{\hat{u}} \oplus \vec{\hat{v}}\}$. The Prover and Verifier must use a common algorithm to compute the same M from $\vec{\hat{u}}$ and $\vec{\hat{v}}$. No interaction is needed for this computation. The check in Step 2 that $\vec{\Psi}$ is the zero vector ensures that $\hat{x}$ is one of the vectors in A. As argued above, if it is not the zero vector, then $\hat{x}_i, \hat{x}_j$ is a random element of T. The vectors $\hat{u}$ and $\hat{v}$ are random, subject to the condition that they generate the subspace A which contains $\hat{x}$.

We now have:

Lemma 3. *Under the QRA, the set* $T = \{01, 10, 11\}$ *is shared-randomness constructible using n random bit commitments with probability of an undetected false proof bounded above by* $(1/2)^n$. *The certificate consists of two n-bit vectors, $n - 2$ numbers modulo N, and two bits. Therefore, it is of length $k(n - 2) + 2n + 2$ bits.*

The error probability for the construction of T has been argued above. It is intuitively reasonable that the procedure is also zero-knowledge under the QRA since then all committed bits are hidden from the Verifier, and the rest of the information in the proof is easily simulated. However, we do not prove zero-knowledge formally here. Instead, we use the procedure for construction of elements in T as a subroutine in our main protocol for satisfiability and prove the entire protocol zero-knowledge in Section 6.

As mentioned before, the lemma also implies that the set T is shared-randomness certifiable. However, we do not make use of this result in this paper. From here on, we concentrate on the techniques and cost of constructing sets related to boolean functions. Later we also introduce techniques for reducing the amortized cost of certifying many pairs of bit commitments as belonging to the set T.

4.2. *The Set $U = \{0111, 1011, 1101, 1110\}$*

If we generate bits $ab \in T$ and $cd \in T$, then $abcd$ may take any value in

$$\{0101, 0110, 0111, 1001, 1010, 1011, 1101, 1110, 1111\}.$$

The set U may be formed by considering only the strings with odd parity. Thus the set U is shared-randomness constructible as follows. The Prover constructs bit commitments x, y along with a proof that $\mathcal{D}(x)\mathcal{D}(y) \in T$. She does the same for bit commitments z, w. Then the Prover proves that $\mathcal{D}(x) \oplus \mathcal{D}(y) \oplus \mathcal{D}(z) \oplus \mathcal{D}(w) = 1$ by displaying a square root of $\mu xyzw$ modulo N. This construction guarantees that $\mathcal{D}(x)\mathcal{D}(y)\mathcal{D}(z)\mathcal{D}(w) \in U$ and, by Lemma 3, fails with probability no more than $2(1/2)^n$. Thus we have proven

Lemma 4. *Under the QRA, the set $U = \{0111, 1011, 1101, 1110\}$ is shared-randomness constructible using $2n$ random bit commitments with probability of an undetected false proof bounded above by $2(1/2)^n$. The certificate consists of $2n - 3$ numbers modulo N, four n-bit vectors, and four bits. Therefore it is of length $k(2n - 3) + 4n + 4$ bits.*

4.3. *All Two-Input Boolean Gates Are Shared-Randomness Constructible Using Trapdoor QR-Bit Commitments*

There are 16 two-input boolean functions $f(x, y)$. If NAND is shared-randomness constructible, then all 16 functions are shared-randomness constructible. Since it will be more efficient to construct gates directly in a circuit (i.e., without simulating the gates using NAND gates), we show how to construct the four most common gates whose truth tables do not form cosets. To do this, use the technique of the previous section to construct bit commitments (a, b, c, d) which commit to a four-bit string in the set U. The following linear transformations map (a, b, c, d) to sets corresponding to boolean gates:

- $S_\wedge = \{000, 010, 100, 111\}$: $(a \oplus c, a \oplus b, b \oplus c \oplus d)$;
- $S_\vee = \{000, 011, 101, 111\}$: $(a \oplus c, a \oplus b, d)$;
- $S_{\overline{\wedge}} = \{001, 011, 101, 110\}$: $(a \oplus c, a \oplus b, a)$;
- $S_{\overline{\vee}} = \{001, 010, 100, 110\}$: $(a \oplus c, a \oplus b, a \oplus b \oplus c)$.

We leave it to the reader to verify that the image of U under each of these transformations is precisely the set of rows of the truth tables for the corresponding boolean function.

There are four more functions whose truth tables do not form cosets ($x \vee \overline{y}$; $\overline{x} \vee y$; $x \wedge \overline{y}$; $\overline{x} \wedge y$). We leave it to the reader to verify that appropriate transformations exist which map U to the truth tables of each of these functions. The following lemma summarizes these observations:

Lemma 5. *Under the QRA, each of the subsets of GF_2^3 corresponding to the rows of the truth tables of two-input boolean functions is constructible. For those functions whose truth tables form cosets, the certificate is constructible without using the shared random strings and with zero probability of an undetected false proof. For those functions whose truth tables do not form cosets, the certificate is shared-randomness constructible using $2n$ random bit commitments for a probability of an undetected bounded above by*

$2(1/2)^n$. *The certificate consists of* $2n - 3$ *numbers modulo* N, *four* n*-bit vectors, and four bits. Therefore it is of length* $k(2n - 3) + 4n + 4$ *bits.*

5. Impossibility Proofs

It would be a powerful and surprising result if we could extend the techniques of Lemmas 1 and 2 to the non-interactive construction of a logically complete set of boolean functions. In this section we present impossibility results related to this issue. We show that it is in fact impossible to extend these techniques. However, this does not show that it is impossible to construct a complete basis for boolean logic without recourse to a shared random string, but it rules out the possibility that it can simply be done with linear maps.

Notation. For any subset $S \subset GF_2^n$, let $S_\mathrm{e}, S_\mathrm{o}$ be the vectors of S with even and odd parity, respectively. For any set S let, $\#S$ be the cardinality of S.

The following are simple properties of GF_2^n.

Fact 1. GF_2^n is a vector space.

Fact 2. Let S be a subspace of GF_2^n. Then either $\#S_\mathrm{o} = 0$ or $\#S_\mathrm{o} = \#S_\mathrm{e} = (\#S)/2$.

Fact 3. Fix $1 \leq i \leq n$ and let S be a subspace of GF_2^n. Then the number of vectors $\vec{v} \in S$ such that $v_i = 1$ is either 0 or $(\#S)/2$.

Lemma 6. *If* R *is a coset, then* $\#\{\vec{v} \mid v_n = 1; \vec{v} \in R\}$ *is either* 0, $(\#R)/2$, *or* $\#R$.

Proof. Let R be induced by a matrix M and a vector $\vec{b} \in GF_2^n$ acting on a subspace S of GF_2^m. If $b_n = 0$, then the lemma follows directly from Fact 3 and the fact that $\{M\vec{x} \mid \vec{x} \in S\}$ is a subspace of GF_2^n (since $(M\vec{x} + \vec{b})_n = (M\vec{x})_n$). Similarly, if $b_n = 1$, then Fact 3 implies that $\#\{\vec{v} \mid v_n = 1; \vec{v} \in R\}$ is either $(\#R)/2$ or $\#R$. □

If λ is a boolean function, we denote by S_λ the set of rows in the truth table for λ. For example, $S_\wedge = \{000, 010, 100, 111\}$.

Lemma 7. *If* λ *is a boolean function on two variables, and if* S_λ *is a coset, then* $\lambda(x, y)$ *is one of the following functions*: $x;\ \neg x;\ y;\ \neg y;\ x \oplus y;\ \neg(x \oplus y);\ 0;\ 1$.

Proof. By Lemma 6, λ must take the value 1 an even number of times. These are the only boolean functions on two variables which have this property. □

Corollary 1. *For all* m, *it is impossible to find a function* $f_\wedge\colon GF_2^m \to GF_2^3$, *using only* XOR *and* NOT, *such that the image of* $f_\wedge$ *is the set* $S_\wedge = \{000, 010, 100, 111\}$.

Corollary 2. *If λ is a boolean function and S_λ is the image of a linear map from any subspace of GF_2^m to GF_2^3, then $\{\lambda, \neg, \oplus\}$ is not a complete basis for boolean logic.*

6. A Non-Interactive Zero-Knowledge Proof of Circuit Satisfiability

Let $\mathcal{C}$ be a circuit with m boolean gates. For simplicity we assume that all gates have one or two inputs. We assume that a Blum integer N for doing trapdoor QR-bit commitments is available, that is, we are not concerned about any precomputation necessary to generate and validate N. Note that values for the security parameters, n and r', implicitly used below will be specified later, as functions of r.

Prover's Algorithm

0. Choose a random non-square μ of Jacobi symbol 1 modulo N. Generate a proof that μ is a non-square. Commit to the values of all inputs to the circuit.
1. Use the techniques of Section 4.3 to construct commitments and certificates to the inputs and output of each non-coset gate.
2. Use the techniques of Lemmas 1 and 2 to compute the values of the outputs of coset gates.
3. For each bit commitment x which is the output of some gate (or an input to the circuit) and corresponds to an input bit commitment y of a non-coset gate, certify that $\mathcal{D}(x) = \mathcal{D}(y)$.
4. Open the bit commitment corresponding to the circuit's output showing that it encodes 1.

Verifier's Algorithm

1. Check that μ has Jacobi symbol 1 modulo N, and verify the proof that μ is a non-square modulo N.
2. Check each gate certification provided by the Prover in Step 1 of the proof.
3. Perform Step 2 of the Prover's construction just as the Prover does.
4. Verify each certification of Step 3 in the proof.
5. Verify the opening in Step 4.

The protocol above can be extended in the obvious way to circuits with more than one output bit.

We now show that it is non-interactive zero-knowledge (under the QRA) in the shared random string model. To show this, one needs to describe a Simulator which can, with an indistinguishable distribution, simulate the joint distribution of shared random string and proof. In our scenario, a Blum integer is assumed to be available before the proof begins. Accordingly, we give a random k-bit Blum integer as input to the Simulator. The only advantage that the Simulator has over the real Prover is that it can create the "shared random string" itself. It will start with a truly random string α, which it will modify to produce a string which is indistinguishable from the original. It will exploit this advantage to use a *square* modulo N as the parameter μ of the commitment scheme. This makes the binding property of the commitment scheme break down, so that the Simulator can now "prove" the theorem without knowing a proof.

Simulator's Algorithm

0. We are given a random k-bit Blum integer N. Write N and a random β with Jacobi symbol -1 modulo N on the transcript. Then initialize a string α' with truly random bits, enough bits to serve as the rest of the shared random string. Finally, choose z at random modulo N and let $\mu = z^2 \bmod N$.
1. The string α is now modified as follows: each k-bit segment representing a number greater than or equal to N is left unchanged. For each k-bit segment of α representing a number less than N, choose x at random modulo N and b, b' as random bits, and replace the segment by the number $x^2\mu^b\beta^{b'} \bmod N$. Let α be the modified string. Output α as the simulation of the shared random string. Note that when numbers are extracted from α, the Simulator knows a square root of every such number, and can "open" them any way it likes when interpreting them as bit commitments.
2. To simulate the proof that μ is a non-square, extract the first r' numbers from α, choose a random non-empty subset W and write W on the transcript. Follow the Verifier's algorithm in computing those products of numbers for which square roots should be supplied, compute these roots (using the fact that roots are known for all numbers occurring), and write them on the transcript.
3. Choose random squares modulo N to play the role of commitments to the inputs of the circuit and write them on the transcript.
4. The rest of the proof consists of a number of applications of the subroutine for constructing (commitments to) members of our set T, and finally a number of openings of commitments to show equality of bits, and one opening to show the circuit outputs 1.

   To simulate construction of members in T, do the following: Choose a random n-bit vector $\vec{\hat{x}}$ and follow the Prover's algorithm to generate from this the vectors $\vec{\hat{u}}$, $\vec{\hat{v}}$. Write them on the transcript and extract the next unused n commitments from α. Compute from this and $\vec{\hat{u}}$, $\vec{\hat{v}}$, following the Verifier's algorithm, the vector of commitments to open, and open them as 0's (using the fact that roots of everything are known). Finally let the number of times the function f should be applied be random from $\{0, 1, 2\}$, and write this number on the transcript. The indices (i, j) are used to extract the pair of commitments to continue with in the following.

   As mentioned, the remaining part of the proof consists only of opening commitments in a way that matches what the Verifier expects. This again is trivially simulated because the roots of all numbers occurring are known.

We claim that under the QRA, this simulation cannot be distinguished from a real proof and shared random string. To see this, assume for contradiction that, for infinitely many k, there exists a satisfiable circuit of size polynomial in k for which the simulation can be distinguished from the real conversation. Then consider the following (non-uniform) algorithm: it gets N, μ as input, where N is a random k-bit Blum integer and μ is random of Jacobi symbol 1 modulo N. It then creates a shared random string, using the same algorithm as in Steps 0 and 1 of the Simulator. This means it knows how to open all commitments in the string. We assume it has the circuit and satisfying assignment hardwired in. Hence it can create a proof simply following the Prover's algorithm. It shows the shared string and proof it has created to the distinguisher and outputs whatever it outputs.

Now just observe that if μ is a non-square modulo N, the distribution shown to the distinguisher is clearly exactly the same as the one output by the honest Prover. On the other hand, if μ is a square, you get exactly the distribution output by the Simulator. So the distinguisher's advantage translates directly to an advantage in guessing the quadratic character of μ. Thus if this advantage is non-negligible in k, we have a contradiction with the QRA.

Except for the part used for validating μ, the expected number of bits the proof uses from the shared random string is at most $4nmk$, where k is the length of the Blum integer N and n is the security parameter used in constructing commitments to members of T (see Section 3.1).

We obtain an upper bound on the length of the non-interactive zero-knowledge proof by assuming that all gates are non-coset gates. Consider the Prover's algorithm. The bits in the proof which are generated at Steps 1 and 3 are no more than $m(k(2n-1)+4n+4)$. The bits which are generated at Steps 0 and 4 are no more than $r'k + r' + 1 + 2km + k$. In a practical scenario, one can expect that the total length of the proof is dominated by the term $2mkn$. In fact, we expect the proofs to be substantially less than $2mkn$ because of the presence of coset gates.

In Section 9.1 we show that n should be $\Theta(\log_2 2m + r)$ and r' should be $\Theta(r)$ to get an overall bound of 2^{-r} on the probability of undetected cheating. Thus, the proof has length $O(mk(\log m + r))$. This is the same asymptotic complexity obtained by the second author [15] and by Kilian and Petrank [26], but gives the best known constants.

7. Gates with More than Two Inputs

Although binary gates are enough to build a circuit to compute any boolean function, using gates with more than two inputs will usually reduce the number of gates in the circuit. This, in turn, will usually make the proof shorter. For example, a three-input MAJORITY gate can be simulated in a straightforward way by five binary gates. However, the cost of constructing such a gate is the same as that of constructing a single AND gate. Here is how:

$M = \{0000, 0010, 0100, 0111, 1000, 1011, 1101, 1111\}$ is the set that must be constructed. Recall the set U from Section 4.2. The set $V = \{abcde \mid abcd \in U; e \in GF_2\}$ is clearly constructible at essentially the same cost as constructing an element from U (except for the extra k-bit string for the commitment to e, or only one bit if this commitment is from the shared random string). Now note that the transformation $abcde \rightarrow (a \oplus c \oplus e, a \oplus b \oplus e, e, b \oplus c \oplus d \oplus e)$ maps V onto M.

Rather than figuring out a direct construction for every gate, however, it may be easier to put some effort into finding a good simulation of gates with many inputs, where by a good simulation, we mean one which contains only negation and binary AND and XOR gates, and has as few AND gates as possible. For example, MAJORITY$(a, b, c) \equiv (a \oplus b) \cdot (a \oplus c) \oplus a$. This formula shows that the cost of constructing such a gate is the same as that of constructing a single AND gate. More generally, a construction due to Muller and Preparata (see Theorem 2.21 in [17]) can be used to compute the majority of n inputs (n odd) using less than n AND gates. Furthermore, combining the latter construction with a result due to Lupanov (see Theorem 4.2 in [17]), we can construct boolean circuits for any symmetric function on n bits using $2n + o(n)$ AND gates. This was improved in [10] to $n + 3\sqrt{n}$ AND gates. In addition, Mirwald and Schnorr [29] have shown that the number of AND gates necessary to compute any quadratic boolean form in n variables is at most $n/2$. Despite the above results, it is fair to say that little is known about the circuit complexity of boolean functions under the "number of AND gates" metric. The applications described in this paper suggest the need for further research on this topic.

8. Non-Interactive Proofs in the Random Oracle Model

In Lemma 5 we showed that constructing input–output bit commitments for any boolean gate (with probability of error less than $2(1/2)^n$) can be done by opening about $2n$ bit commitments and sending about $4n$ bits. The purpose of opening the $2n$ bit commitments is to show that they are all 0. Step 3 of the Prover's and Verifier's algorithm (in Section 6) also amounts to proving that a set of bit commitments are 0. Thus it is natural to ask whether there is a more efficient way to prove a set of bit commitments are 0 without actually opening all of them. This can indeed be done under the random oracle model, in which the players do not have advance knowledge of the shared random bits. The opening of bit commitments at Steps 1 and 3 can be replaced by a fifth step of the protocol in which the Prover *probabilistically* shows that all these bit commitments (stored by both Prover and Verifier in an array denoted by *COMM_0*) are in fact 0. This can be done as follows: if not all bit commitments are 0, then the exclusive-or of a random subset (where a random subset is chosen by independently choosing each commitment with probability $1/2$) of the bit commitments is 1 with probability $1/2$. A commitment to the exclusive-or of such a subset can be non-interactively computed by multiplying modulo N the commitments to the members of the subset. Thus the Verifier can be convinced that all bit commitments are 0, with confidence level $1 - (1/2)^{r'}$, by the Prover opening r' non-interactively computed bit commitments to the exclusive-ors of randomly chosen subsets. However, one more query to the randomness source is needed to select r' random subsets of the bit commitments.[6]

The previous observations are summarized in the following algorithm. We leave to the reader the details of the corresponding Verifier's algorithm.

Prover's Algorithm

0. The random oracle is queried for the first time.
1. Choose and validate the parameter μ for the commitment scheme. Commit to the values of all inputs to the circuit. The commitments are obtained from the oracle.
2. Use the techniques of Section 4.3 to construct commitments and certificates to the inputs and output of each non-coset gate. However, do not open any bit commitments at this step. Instead, store those commitments in an array *COMM_0*.

[6] The security parameter r' used here can be the same as the r' used in showing that μ is a non-square.

3. Use the techniques of Lemmas 1 and 2 to compute the values of the outputs of coset gates.
4. For each bit commitment x which is the output of some gate (or an input to the circuit) and corresponds to an input bit commitment z of a non-coset gate, add the commitment $x \cdot z \bmod N$ to the array *COMM_0*.
5. The random oracle is queried for the second time.
6. Use the bits obtained from the oracle to choose r' random subsets of the elements in *COMM_0*. Multiply the elements of each of these subsets modulo N. Open the resulting bit commitment to show it encodes 0.
7. Open the bit commitment corresponding to the circuit's output showing that it encodes 1.

Since there are m gates in the circuit, the Prover need send at most $4nm$ bits at Step 2 plus r' bit commitment openings at Steps 1 and 6 to prove circuit satisfiability with negligible probability of getting away with a false proof. This is an extremely short proof, and it is zero-knowledge in the random oracle model.

9. Non-Interactive Discreet Proofs Obtained Using the Random Oracle Heuristic

It is a standard technique in cryptographic protocol design to substitute a random "challenge string" in a provably secure protocol by a string which is the output of a hash function on an appropriately chosen input. This idea is originally due to Blum [3]. Some well-known applications of this technique occur in Fiat–Shamir's scheme [19], in Schnorr's scheme [32], and in the DSA [31]. Bellare and Rogaway, in [1], formalize the technique and give some basic guidelines for choosing the hash function. They also coin the term "random oracle heuristic" to refer to this technique. We refer the reader to the latter paper. We stress only that it is crucial for the security of the discreet proof that the bits generated when simulating the second query to the random oracle depend on the bits of the proof generated in the previous steps.

An analysis of the length and error probability of our proofs follows.

9.1. *Performance*

Until now we have been assuming the worse case scenario: that all gates in the circuit are non-coset gates. In fact, it has been recently shown that the number of conjunctions necessary to implement a boolean function is likely to be significantly smaller than the total number of gates. Given a boolean function f, denote by $c_\wedge(f)$ the minimum number of AND gates necessary to construct a circuit for f using only gates in $\{\oplus, \neg, \wedge\}$. This is called the "conjunctive complexity" of f. In [10] it is shown that a random function f on n bits will, with probability exponentially close to 1, have conjunctive complexity $c_\wedge(f) < 2^{n/2+1}$. In contrast, it is known that the number of total gates is $\Theta(2^n n^{-1})$ for all but a negligible fraction of boolean functions on n bits. Although all NP functions lie in this "negligible fraction," the above results are a strong indication that the conjunctive complexity of any function is likely to be considerably smaller than its boolean complexity. This implies that the number of non-cosets gates necessary for implementing functions is also likely to be considerably smaller than the total number of gates. Therefore we introduce a new parameter θ to denote the number of non-coset gates of the circuit. (The reader may choose to think of the circuit as containing only AND, XOR, and NOT gates. In this case θ is just the number of AND gates.)

The length of the proof is essentially $4n\theta + r'k$ bits, where r' is the number of random subsets of bit commitments opened at the final step of the proof. The parameters n and r' determine the probability that the Prover can get away with a false proof.

There are three events which *might* allow the Prover to cheat:[7]

1. In the process of constructing the commitments to a gate, a vector $\hat{x}$ derived from the random source is the zero vector.
2. In the final step of the proof, not all bit commitments are 0, yet the exclusive-or of the r random subsets of the bit commitments are all 0.
3. The numbers from the random string used in validating that μ is a non-square all have the same quadratic character.

The probabilities of the second and third event are $2^{-r'}$, regardless of θ, but the probability of the first event depends on θ. For the first event, there are 2θ vectors to worry about. The probability that at least one of these vectors is 0 is $1 - (1 - (1/2)^n)^{2\theta}$. Letting $n = \log_2(2\theta) + r'$, we have $1 - (1 - (1/2)^n)^{2\theta} \longrightarrow 2^{-r'}$. The convergence is fast.[8] We choose r' to be $r + 2$ to obtain a total probability of unsuccessful cheating of at most 2^{-r}. Thus, the length of the proof in bits is (omitting lower-order terms)

$$4n\theta + 2rk = 4\theta(\log_2(2\theta) + r) + 2rk = 4\theta \log_2(2\theta) + r(4\theta + 2k). \qquad \text{(I)}$$

In other words, the discreet proof has length $O(\theta(\log\theta + r) + kr)$ to achieve an upper bound of 2^{-r} on the probability of successful cheating by the Prover.

Having seen these results, it should be noted that the random oracle heuristic can be applied to any *interactive* zero-knowledge proof where the Verifier sends only random bits. Thus a comparison with other known techniques is in order. Some of the most efficient such techniques are found in [14], resp. [16], where zero-knowledge proofs are given based on so-called q-one-way homomorphisms, resp. collision-intractable hash functions. These methods can be applied to general Boolean circuits. The resulting lengths of proofs obtained after applying the heuristic are not directly comparable with ours, because they depend on the total complexity of the function involved: for functions with low conjunctive complexity, our proofs can be much shorter, while for functions with conjunctive complexity equal to the total complexity, our proofs are longer by an additive contribution of roughly $m \log m$. Another efficient technique can be found in [20]. It is based on the strong RSA assumption, and produces *extremely* short proofs of statements that can be specified efficiently using large integer arithmetic. It does not seem to help at all, however, for general Boolean functions.

[7] In practice even if one of these events occurs, the Prover is unlikely to be able to cheat.

[8] For example, for $\theta = 2000$ and $r = 50$ the estimate 2^{-50} is accurate up to 29 decimal digits.

address x_1 x_0	entry b_2 b_1 b_0
0 0	1 1 0
0 1	0 1 0
1 0	1 1 1
1 1	0 0 1

$$b_2 = \bar{x}_1\bar{x}_0 \oplus x_1\bar{x}_0$$
$$b_1 = \bar{x}_1\bar{x}_0 \oplus \bar{x}_1 x_0 \oplus x_1\bar{x}_0$$
$$b_0 = x_1\bar{x}_0 \oplus x_1 x_0$$

Fig. 2. A 4 $\times$ 3 table and corresponding output equations.

10. The Conjunctive Complexity of Substitution-Permutation Ciphers

Substitution-permutation ciphers, such as DES, are widely used in cryptography. In this section we consider the question of how many AND gates are necessary for the construction of such circuits. Our aim is to minimize the number of AND gates so as to shorten the discreet non-interactive proofs for such circuits.

In permutation-substitution ciphers, AND gates are only necessary for table indexing. We assume the tables which need to be indexed into are public (for example, such is the case with DES). Given a table of size $m \times 2^n$ (n input bits determine m output bits), we need to produce all 2^n minterms on n boolean variables. Once we have done that, the completion of the table indexing circuit will require only additional XOR gates. This is true because in order to produce one output bit, one can simply take the XOR of all those minterms which could produce an output of 1. At most one of those minterms will have the value 1. If one does, the result will be 1; otherwise, it will be 0 (an example is depicted in Fig. 2).

It is possible to use fewer AND gates than the most straightforward construction would indicate. For example, consider the case $n = 2$. The following equations show that only one AND gate is needed (to compute xy):

$$\begin{aligned} x\bar{y} &= xy \oplus x, \\ \bar{x}y &= xy \oplus y, \\ \bar{x}\bar{y} &= \bar{x}y \oplus \bar{x}. \end{aligned}$$

More generally, we can prove the following theorem:

Theorem 1. *Given the basis* $\{\oplus, \neg, \wedge\}$, *only* $2^n - (n+1)$ AND *gates are sufficient for computing all minterms on n boolean variables.*

Proof. Our proof will be constructive. The argument for the number of gates will be a double induction. For $n = 1$, only the variable and its complement need to be produced, so no AND gates are necessary. Since $2^1 - (1 + 1) = 0$, the theorem holds for $n = 1$.

Construction

Assume all minterms (of sizes 1 through $m-1$) have been constructed containing boolean variables $a_1 \cdots a_{m-1}$. Assume also that the number of AND gates used is $2^{m-1}-(m-1+1)$. Call a minterm "positive" if none of the variables are negated. There are $2^{m-1}-1$ positive minterms involving variables $a_1 \cdots a_{m-1}$. We AND each of these minterms with a_m.

We claim that it is now possible to construct all minterms on the variables $a_1 \cdots a_m$ using only XOR gates. We show this by induction on the number of negated variables in the target minterm. If there are no negated variables, we are done. Inductively assume that we have constructed all minterms which have at most l negated variables. It will be enough to show that we can negate an arbitrary variable of any of these minterms. Without loss of generality, suppose the minterm is $X = \overline{a_1} \cdots \overline{a_l} a_{l+1} \cdots a_k$. We wish to negate a_{l+1}. By the inductive hypothesis we have already constructed the minterm $Y = \overline{a_1} \cdots \overline{a_l} a_{l+2} \cdots a_k$. Now simply observe that $X \oplus Y = \overline{a_1} \cdots \overline{a_l}\, \overline{a_{l+1}} a_{l+2} \cdots a_k$.

The number of AND gates in this construction is $2^{m-1} - (m - 1 + 1) + 2^{m-1} - 1 = 2^m - (m + 1)$ as claimed. □

11. Length of Discreet Proofs for RSA and DES

The aim of this section is to show that our proofs are short enough to be used in practice for commonly used cryptographic functions. No major attempt at circuit optimization has been done. That is the subject of work in progress. We use only $\oplus$, $\neg$, and $\wedge$ gates. We ignore here the cost of validating the commitment scheme parameter μ because it can be validated once and for all and reused several times.

Suppose we want to prove to a Verifier that we know a DES key K such that $\mathrm{DES}_K(X) = Y$ for public X and Y.[9] The techniques introduced in this paper involve constructing a circuit for DES where the unknown input is the key K and the output is $\mathrm{DES}_K(X)$. Since Y is public, all the output bits of the circuit will be opened so the Verifier can check that the output is Y. The circuit must be constructed so as to minimize the number of AND gates. In DES, AND gates are only needed for indexing into the S-boxes. A straight-forward construction using Theorem 1, but not exploiting any structure in the S-boxes (there should not be any, according to unofficial statements on the matter by NSA and IBM), yields a circuit with 57 AND gates per S-box. There are 8 S-boxes and each is used 16 times. Therefore the number of AND gates in our circuit is 7296. It is currently assumed that the QRA holds for Blum integers of size 1024 bits. With this security parameter, (I), Section 9.1, tells us the length of the discreet proof is

$$4\theta \log_2(2\theta) + r(4\theta + 1024) < 403{,}700 + 30{,}208r \quad \text{bits.} \tag{II}$$

As for the parameter r, this depends on the security level required by the application. For $r = 10$, (II) yields a proof of length under $90KB$.[10] The corresponding length for $r = 40$ is under $200KB$.

We now consider proving that we know a decryption key d for a public RSA key (N, e). Since knowing d is random polynomial time equivalent to knowing the factorization of

[9] Note that this was the technique used by the original UNIX password scheme (with $X = 0$) [30]. In that scheme, the password must be transmitted in the cleartext. Consider the vulnerability of remote logins under this scheme, something which is now done on a routine basis.

[10] Note that $r = 10$ does not imply that the Prover can cheat with probability close to .001. That would be the case only if the Prover, without knowing K, could somehow make the output bits into Y by cheating at only one gate. Although a scenario like this one could be constructed, it is highly unlikely to occur in practice. It is beyond the scope of this paper to attempt to answer the question of how can circuits be constructed so that discreet proofs are resilient to failure (undetected false proofs) up to a threshold value of t gates.

N, we can prove this fact instead. Therefore all we need is a circuit which multiplies two inputs P and Q. For simplicity we assume both P and Q are of length 512 bits. In this case, the Prover need only open the output bits of the multiplication circuit. The Verifier checks that the output is N.

Let $mult_\wedge(a)$ denote the conjunctive complexity of multiplication of two integers of length a. The number of conjunctions in a multiplication circuit using Karatsuba–Ofman [24] is closely approximated by the recurrence $mult_\wedge(n) \leq 3mult_\wedge(\lceil n/2 \rceil) + 9\lceil n/2 \rceil$.[11] Using the base $mult_\wedge(1) = 1$ this recurrence yields $mult_\wedge(512) \leq 192{,}222$. For $r = 10$ we get

$$4\theta \log_2(2\theta) + r(4\theta + 1024) < 2.7MB.$$

The corresponding value for $r = 40$ is under $5.5MB$.

It should be possible to do significantly better than this. We leave that as a very interesting open problem.

Acknowledgments

We would like to thank Laila Mathiassen for her help in calculating the number of AND gates in a circuit for multiplying two integers. We thank Denis Pochuev for helping us with some of the circuit complexity issues raised by this research. We thank Dennis Meilicke for many discussions and much helpful information. We also thank the anonymous referees for comments which have greatly helped in improving the exposition.

References

[1] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings First Annual Conference on Computer and Communications Security* (1993).

[2] Blum, M.: Coin flipping by telephone. In *IEEE COMPCON* (1982), pp. 133–137.

[3] Blum, M.: Personal communication with Rene Peralta, 1983.

[4] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In *Proceedings of the* 20*th Annual ACM Symposium on the Theory of Computing* (1988), pp. 103–112.

[5] Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge. *SIAM Journal on Computing*, **20** (1991), 1084–1118.

[6] Boyar, J., Brassard, G., Peralta, R.: Subquadratic zero-knowledge. *Journal of the Association for Computing Machinery*, **42** (1995), 1169–1193.

[7] Boyar, J., Krentel, M., Kurtz, S.: A discrete logarithm implementation of zero-knowledge blobs. *Journal of Cryptology*, **2** (1990), 63–76.

[8] Boyar, J., Lund, C., Peralta, R.: On the communication complexity of zero-knowledge proofs. *Journal of Cryptology*, **6** (1993), 65–85.

[9] Boyar, J., Peralta, R.: Short discreet proofs. In *Advances in Cryptology—Proceedings of EUROCRYPT* 96 (1996), vol. 1070 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 131–142.

[10] Boyar, J., Peralta, R., Pochuev, D.: On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. *Theoretical Computer Science*, **235** (2000), 43–57.

[11] Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, **37** (1988), 156–189.

[11] To derive this bound, we have used the fact that the multiplicative complexity of adding two n-bit numbers is no more than n. This is a consequence of MAJORITY, the "majority" function on three input bits, having conjunctive complexity 1 (see Section 7 or [10]).

[12] Brassard, G., Crépeau, C.: Zero-knowledge simulation of boolean circuits. In *Advances in Cryptology—Proceedings of CRYPTO* 86 (1987), vol. 263 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 223–233.

[13] Cramer, R., Damgård, I.: Linear zero-knowledge: a note on efficient zero-knowledge proofs and arguments. In *Proceedings of the* 29*th Annual ACM Symposium on the Theory of Computing* (1997), pp. 436–445.

[14] Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or can zero-knowledge be for free? In *Advances in Cryptology—Proceedings of CRYPTO* 98 (1998), vol. 1462 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 424–441.

[15] Damgård, I.: Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *Advances in Cryptology—Proceedings of EUROCRYPT* 92 (1993), vol. 658 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 341–355.

[16] Damgård, I., Pfitzmann, B.: Sequential iteration of interactive arguments and an efficient zero-knowledge argument for NP. In *Automata, Languages and Programming*. 25*th International Colloquium, ICALP* '98 (1998), vol. 1443 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 772–783.

[17] Dunne, P.: The Complexity of Boolean Networks. Academic Press, New York, 1988.

[18] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero-knowledge proofs based on a single random string. In *Proceedings of the* 31*st IEEE Symposium on Foundations of Computer Science* (1990), pp. 308–317.

[19] Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology—Proceedings of CRYPTO* 86 (1987), vol. 263 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 186–194.

[20] Fujisaki, E., Okamoto, T.: Statistical zero-knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology—Proceedings of CRYPTO* 97 (1997), Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 16–30.

[21] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing Machinery*, **38** (1991), 691–729.

[22] Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences*, **28** (1984), 270–299.

[23] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, **18** (1989), 186–208.

[24] Karatsuba, A., Offman, Y.: Multiplication of multidigit numbers on automata. *Doklady Akademii Nauk SSSR*, **145** (1962), 293–294.

[25] Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the* 24*th Annual ACM Symposium on the Theory of Computing* (1992), pp. 723–732.

[26] Kilian, J., Petrank, E.: An efficient non-interactive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, **11** (1998), 1–27.

[27] De Santis, A., Di Crescenzo, G., Persiano, G.: Secret sharing and perfect zero knowledge. In *Advances in Cryptology—Proceedings of CRYPTO* 93 (1993), vol. 773 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 73–84.

[28] De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge with preprocessing. In *Advances in Cryptology—Proceedings of CRYPTO* 88 (1989), vol. 403 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 269–282.

[29] Mirwald, R., Schnorr, C.: The multiplicative complexity of quadratic boolean forms. *Theoretical Computer Science*, **102** (1992), 307–328.

[30] Morris, R., Thompson, K.: Password security: a case history. *Communications of the ACM*, **22** (1979), 594–597.

[31] NIST: Digital signature standard. NIST FIPS PUB 186, 1994.

[32] Schnorr, C.: Efficient signature generation for smart cards. *Journal of Cryptology*, **4** (1991), 161–174.

[33] van de Graaf, J., Peralta, R.: A simple and secure way to show the validity of your public key. In *Advances in Cryptology—Proceedings of CRYPTO* 87 (1988), vol. 293 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 128–134.