BRICS

Basic Research in Computer Science

BRICS RS-97-27 Cramer & Damgård: Zero-Knowledge Proofs for Finite Field Arithmetic

Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?

Ronald Cramer
Ivan B. Damgård

BRICS Report Series **RS-97-27**

ISSN 0909-0878 **November 1997**

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/27/`

Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?

Ronald Cramer [1] and Ivan Damgård [2]

Abstract

We present zero-knowledge proofs and arguments for arithmetic circuits over finite prime fields, namely given a circuit, show in zero-knowledge that inputs can be selected leading to a given output. For a field $GF(q)$, where q is an n-bit prime, a circuit of size $O(n)$, and error probability 2^{-n}, our protocols require communication of $O(n^2)$ bits. This is the same worst-cast complexity as the trivial (non zero-knowledge) interactive proof where the prover just reveals the input values. If the circuit involves n multiplications, the best previously known methods would in general require communication of $\Omega(n^3 \log n)$ bits.

Variations of the technique behind these protocols lead to other interesting applications. We first look at the Boolean Circuit Satisfiability problem and give zero-knowledge proofs and arguments for a circuit of size n and error probability 2^{-n} in which there is an interactive preprocessing phase requiring communication of $O(n^2)$ bits. In this phase, the statement to be proved later need not be known. Later the prover can *non-interactively* prove any circuit he wants, i.e. by sending only one message, of size $O(n)$ bits.

As a second application, we show that Shamirs (Shens) interactive proof system for the (IP-complete) QBF problem can be transformed to a zero-knowledge proof system with the same asymptotic communication complexity and number of rounds.

The security of our protocols can be based on any one-way group homomorphism with a particular set of properties. We give examples of special assumptions sufficient for this, including: the RSA assumption, hardness of discrete log in a prime order group, and polynomial security of Diffie-Hellman encryption.

We note that the constants involved in our asymptotic complexities are small enough for our protocols to be practical with realistic choices of parameters.

[1] ETH Zurich, cramer@inf.ethz.ch

[2] Aarhus University, BRICS (Basic Research in Computer Science, center of the Danish National Research Foundation), ivan@daimi.aau.dk

1 Introduction

Zero-Knowledge interactive proofs [17] and arguments [5] allow a prover to convince a verifier that a statement (on membership in a language) is true while revealing nothing but the validity of the assertion.

Interactive proofs are secure against cheating even by infinitely powerful provers, on the other hand, zero-knowledge can - at least for NP-hard problems - only be guaranteed relative to a computational assumption (unless the polynomial time hierachy collapses, [13]). The first zero-knowledge interactive proof for an NP-hard problem was given in [16], this was later extended to build zero-knowledge proofs for all languages in IP[6], the class of languages with interactive proofs[3].

Interactive arguments are only secure against polynomial time provers, and so require computational assumptions to establish soundness. On the other hand, they can provide perfect (unconditional) zero-knowledge for all of NP, as shown in [5].

Summarizing informally, these basic results say that, under reasonable computational assumptions, all languages that have an interactive proof (argument), also have a zero-knowledge interactive proof (argument), albeit a much less efficient one. From this has emerged naturally a line of research aimed at improving the efficiency (in terms of communication complexity) of zero-knowledge protocols for NP complete problems such as SAT [4, 18, 19, 9]. It is natural to ask to what extent we can reach the optimal situation, where giving a zero-knowledge interactive proof for SAT, or other problems in IP, is as efficient as giving a mere interactive proof, in other words, can zero-knowledge be for free? In this paper we give protocols showing that in some cases, zero-knowledge may indeed be almost or entirely for free.

We first present zero-knowledge proofs and arguments for arithmetic circuits over finite prime fields, namely given a circuit with multiplication and addition gates, show in zero-knowledge that inputs can be selected leading to a given output. We will refer to this as the *arithmetic circuit problem.* For a field $GF(q)$, where q is an n-bit prime, a circuit of size $O(n)$, cryptographic security parameter n and error probability 2^{-n}, our protocols require communication of $O(n^2)$ bits. For interactive proof systems capable of handling any arithmetic circuit, we believe this is an optimal result: the simplest non-zero knowledge proof system would be to just reveal the inputs, which may cost $\Omega(n^2)$ bits.

If the circuit involves n multiplications, the best previously known method is to

[3] In fact, IP is equal to PSPACE, as shown by Shamir[23]

rewrite the multiplications to Boolean circuits and use the best known protocol for circuit satisfiability. This leads to a communication complexity of $\Omega(n^3 \log n)$ bits. As a more precise account of the performance of our protocol, we mention that its communication complexity is $O((m+t)(l+n)\lceil k/n \rceil)$ bits where the error probability is 2^{-k}, l is the cryptographic security parameter, m is the number of inputs and t is the number of multiplication gates. Thus linear operations are essentially for free.

So for arithmetic circuits, it seems the only price we must pay for zero-knowledge is the interaction required. For an NP hard problem, this cannot be avoided unless $NP \subset BPP$. But we can partially avoid it by going to the model of non-interactive proofs or arguments with preprocessing [25]. In this model, we present protocols for the Arithmetic Circuit Problem and Boolean Circuit Satisfiability. Here, the prover and verifier are allowed to do an interactive preprocessing stage, in which it is not necessary to know which statement (circuit) will be proved later (except perhaps for an upper bound on its size). Then, at a later time, the prover should be able to prove any circuit of his choice by sending only one message.

For the arithmetic circuit problem, the complexity of both our preprocessing and proof phase is $O(n^2)$ bits (the same as for the interactive protocol mentioned above).

For the Boolean circuit satisfiability problem using a circuit of size n, cryptographic security parameter n and error probability 2^{-n}, our preprocessing has size $O(n^2)$ bits, whereas the proof is of size $O(n)$ bits. Thus the proof stage has the same worst case complexity as the obvious interactive proof for SAT, where one just sends a satisfying assignment, which can in general have size $\Omega(n)$. Since it is not known how to make do with less than this for an interactive proof for SAT and given that the interaction (in the preprocessing) cannot be avoided unless $NP \subset BPP$, our result seems close to optimal. We also note that our *total* communication complexity is the same as that of the best previously known zero-knowledge interactive proofs [9] (which could not be split in a preprocessing and proof phase).

To compare with earlier work on interactive arguments, we need to give a more precise account of the performance of our protocols: for an error probability of 2^{-k}, and cryptographic security parameter l, the complexity of the preprocessing is $O(n)max(k,l)$ bits in the proof case, and $O(ln+k)$ bits in the argument case. The proof phase has size $O(n+l)$ bits in both cases. The best earlier work on arguments is by Cramer and Damgård [9] who obtained $O(n)max(l,k)$, and by Kilian [19] who obtained $O(kl \log l)$. None of these protocol could be split in a preprocessing and proof phase, as ours. Our total complexity improves on [9] and is not directly comparable to [19]. It is superior to [19] for some choices of parameters, e.g. when all parameters are chosen equal to n, but inferior in other cases - in particular because of the very interesting fact that the result from [19] does not depend on n.

From a practical point of view, Kilian's results are not of much relevance, since they are based on PCP's [2], and hence rely on the elaborate reductions needed to build PCP's. By contrast, the constants involved in our asymptotic complexities are small enough for our protocols to be practical with realistic choices of parameters. For example, our most efficient argument for SAT based on RSA produces a proof stage of size $2(n + l)$ bits, where l is the length of the RSA modulus used. Moreover, we believe that non-interactive protocols with preprocessing and small proofs have significant advantages over ordinary interactive protocols: In real networks, it is often the case that large amounts of bandwidth is available at low prices during particular time intervals, typically at times where the network operator expects traffic to be low. The preprocessing can then be done at such times, which makes the added cost of later doing a proof almost negligible: the prover must in any case send a message describing the circuit he wants to prove satisfiable, and appending our proof makes this message larger by only a constant factor.

Our final result shows that Shamirs (Shens) [23, 24]interactive proof system for the (IP-complete) QBF problem can be transformed to a zero-knowledge proof system with the same asymptotic communication and round complexity [4]. Thus for QBF, zero-knowledge may in fact be entirely for free.

The security of our protocols can be based on any one-way group homomorphism with a particular set of properties. We give examples of special assumptions sufficient for this, including: the RSA assumption, hardness of discrete log in a prime order group, and polynomial security of Diffie-Hellman encryption. Our main technical tool is a method for building from the homomorphisms assumed a commitment scheme, where commitments can contain elements from a finite prime field, and where multiplication and comparison of committed values can be handled very efficiently.

[4] It is, of course, well known [6] that it is *possible* to build a zero-knowledge protocol from Shen's or Shamir's proof systems, provided one-way functions exist. However, the transformation from [6] leads a huge loss of efficiency.

2 Protocol Descriptions

Our basic protocols make use of a commitment scheme for numbers modulo q, for some prime q. This section describes these protocols in a way that is independent from any particular implementation of the commitment scheme. We will describe how to build honest verifier zero-knowledge protocols. Standard techniques may then be used to make protocols that are zero-knowledge in general.

2.1 Notation and Properties for Commitments

For now, the reader may think of the commitment scheme intuitively as follows: the prover P puts an integer a into a closed box, where $0 \leq a < q$ for some fixed prime q and gives it to the verifier V. At this point, V cannot open the box, and P cannot change his mind about a. However, P may later choose to open a box and reveal the contents to V. More details on commitments can be found in Section 3.

In a real implementation, commitments will be represented by bit strings. We will use l to denote the length of a commitment, and we will assume that to open a commitment, it suffices to send, in addition to the value revealed, a string of length at most l bits.

We will need the following properties:

1. From commitment A containing a, resp. B containing b, V can on his own compute a commitment containing $a+b \bmod q$, or he may choose to compute one containing $a - b \bmod q$. Since in our concrete examples, commitments are in a multiplicative group, we will denote these commitments by $A \cdot B$, resp. AB^{-1}. The property also implies that V can multiply or add constants into a commitment. We will let A^c, cA, cA^{-1} denote commitments to $ca, c+a, c-a \bmod q$, as computed from A.

2. There is a protocol by which P can convince V in honest verifier zero-knowledge that a given commitment is a *bit commitment*, i.e. P knows how to open it to reveal 0 or 1.

3. There is a protocol by which P can convince V in honest verifier zero-knowledge that he knows how to open a set of given commitments A, B, C to reveal values a, b, c, for which $c = ab \bmod q$. In particular, this means that P can show that he knows how to open a single commitment A (by choosing $C = A$ and B a default commitment to 1).

In some implementations of commitments, q can be chosen independently of l, we then talk about commitments with *unbounded q*. In other implementations, q must be $2^{O(l)}$. In such cases with bounded q, we will assume that $q = 2^{\delta l}$, for some constant $\delta > 0$.

Property 1 above may be used to show relations on committed bits. Concretely, suppose we want to show for two sets of bit-commitments $D_0, ..., D_n$ and $C_0, ..., C_n$, where $n < \log q$, that the same bit b_i is contained in C_i and D_i, for $i = 1...n$. This can be done much more efficiently than by comparing each C_i, D_i individually. For this, we have the following protocol:

EQUALITY PROTOCOL

1. the verifier first computes the commitments $C = C_n^{2^n} \cdot C_{n-1}^{2^{n-1}} \cdot .. \cdot C_0$, and $D = D_n^{2^n} \cdot D_{n-1}^{2^{n-1}} \cdot .. \cdot D_0$ which should both be commitments to the number whose binary representation is $b_n b_{n-1} ... b_0$.
2. Finally prover and verifier compute CD^{-1} and the prover opens the result to reveal 0.

It is easy to see that this game reveals nothing about the value of $b_0, ..b_n$, and that if P can open each of the commitments to reveal a one-bit value, all pairs C_i, D_i contain the same bit, or he can break the commitment scheme.

2.2 Protocols for Arithmetic Circuits over $GF(q)$

In this section, we are given an arithmetic circuit Ψ over $GF(q)$, where q is an n-bit prime, containing gates $G_1, .., G_v$, where we assume that G_v is the gate computing the final output from the circuit. We assume for simplicity that there is only one output value computed, we are given a value y for this output, and the prover's goal is to demonstrate that inputs can be selected that lead to output y.

All gates have fan-in at most two and arbitrary fan-out. Gates may be multiplication gates, addition gates, and addition or multiplication by a constant.

The protocol takes place in a series of steps:

STEP 0
The prover and verifier go through the setup phase for the commitment scheme, as described in Section 3. This can be done once and for all, and the instance of the commitment scheme generated can be reused in several protocol executions.

STEP 1
The prover makes m commitments $I_1, .., I_m$, such that I_j contains input value $x_j \in GF(q)$. The input values are selected such that the circuit computes y as output. The prover also makes t commitments $T_1, ..., T_t$, such that T_i contains the value that is output by the i'th multiplication gate in the circuit, given that the inputs are $x_1, ..., x_m$. All commitments produced are sent to V, and P proves that he knows how to open all of them (using the third assumed property of commitments).

STEP 2
Both P and V compute, based on $I_1, .., I_m, T_1, .., T_t$ and using the first assumed property of commitments, for each gate commitment(s) representing its input value(s), and a commitment representing its output value.

PROOF, Step 3
For each multiplication gate, do the following: let A, B be the commitments representing the input values a, b, and let C be the commitment representing the output value c. P uses the third property of commitments to convince V that $ab \bmod q = c$.

PROOF, Step 4
P opens the commitment representing the output value of G_v.

V accepts, if and only if all proofs in Steps 1 and 3 are accepted, and P correctly opens the commitment in Step 4 to reveal y.

The following is immediate from inspection of the protocol description:

Lemma 2.1 *If inputs for Ψ can be selected leading to output y and P follows the protocol, then V always accepts. The communication complexity of the protocol is $(m + t + 1)l + \beta(m, t, l, k, n)$ bits, where $\beta(m, t, l, k, n)$ is the communication complexity for doing all the interactive proofs required in Steps 1 and 3.*

2.2.1 A Non-interactive with Preprocessing Variant

We sketch here a variant of the arithmetic circuit protocol that is non-interactive with preprocessing. The asymtotic complexity for the preprocessing is the same as the original protocol, whereas the proof phase has complexity $O((m+t)(l+n))$ bits. The variant is based on a technique borrowed from Beaver et al. [1].

In the preprocessing, the prover will produce commitments $J_1, ..., J_m$ containing random values, and t triples of commitments of form D, E, F containing random values d, e, f such that $de = f \bmod q$. The prover will show that he can open all commitments and that the multiplicative relations hold.

In the proof phase, a circuit with input values is known to the prover. Consider a fixed multiplication gate. It is first assigned a distinct triple D, E, F from the preprocessing. Let a, b, c, where $ab = c \bmod q$ be the values actually occurring at the gate. The prover can now send to the verifier $\epsilon = a - d$ and $\delta = b - e$. Now, the verifier can on his own compute a triple A, B, C containing a, b, c by letting $A = \epsilon D$, $B = \delta E$ and $C = \epsilon\delta F \cdot D^{\delta} \cdot E^{\epsilon}$.

In the same way, the prover tells the verifier how to modify the J_i's to get commitments containing the correct inputs to the circuit by giving the differences between the random values in the J_i's and the actual values.

All that remains is for the prover to show that "gates connect correctly", i.e. that if e.g. A' represents the output from one gate, which is connected to the input of another gate, represented by A, the prover shows that A and A' contain the same value by opening $A'A^{-1}$ as 0.

2.3 Non-Interactive Protocols with Preprocessing for SAT

For the protocol description, we first need some notation and definitions: We will assume (without loss of generality) that the circuit to be proved satisfiable later is given with at most n NAND gates with fan-in 2 and arbitrary fan-out.

Definition 2.2 *A* NAND-Table *is a matrix with 4 rows and 3 columns containing commitments. A NAND-table is* correct*, if any of its rows A, B, C satisfies that the prover can open the three commitments to reveal bits a, b, c, where $a \wedge b = \neg c$. An NAND table is* useful *if it is correct, and if one obtains, by opening all its commitments and permuting the rows, the truthtable of the NAND-function.*

In the following the honest prover will make only useful NAND-tables, but to keep the prover from cheating it will be enough to force him to generate at least correct NAND-tables.

To show correctness of a NAND-table, P can first show that the 8 commitments in the two first positions of each row are bit commitments, by the second assumption on commitments. Then for each row A, B, C, containing a, b, c, P uses properties 1 and 3 above to show that $1 - c = ab \bmod q$. Assuming that a and b are 0/1 values, this ensures that so is c, and that $\neg c = a \wedge b$.

We are now ready to start giving the protocol in detail. First is:

STEP 0

The prover and verifier go through the setup phase for the commitment scheme, as described in Section 3. This can be done once and for all, and the instance of the commitment scheme generated can be reused in several protocol executions.

PREPROCESSING
The prover makes n useful NAND-tables, using for each table an independently and uniformly chosen permutation of the rows. He proves that all NAND-tables are correct, as described above.

For the proof phase, we are given the concrete circuit Φ that should be shown to be satisfiable, containing gates $G_1, .., G_n$, where we assume that G_n is the gate computing the final output from the circuit. The proof string to be sent to V is constructed by P as follows:

PROOF, Step 1
For $i = 1..n$, take the first unused NAND table T_i from the preprocessing and assign it to gate G_i.

PROOF, Step 2
Fix a set of input bits that satisfy the circuit. For each $i = 1...m$, P selects a row in T_i such that this row contains the 2 input bits and the output bit of G_i in a computation on the satisfying input. P includes 2 bits in the proof string indicating which row is selected.

By selecting rows in all truth tables, P has essentially defined a computation in the circuit. He must now show that this computation is consistent, by demonstrating that the output from one gate equals the input to another gate, if a wire connects them, and also that if the same input bit is used in several gates, the same value for this bit is consistently used.

PROOF, Step 3
Consider any wire W in the circuit. We will associate a pair of commitments to W as described in the algorithm below. If W connects an input bit to a gate in the circuit, we will, for convenience in the description of the algorithm, associate a commitment Y_W to W. This commitment is defined during execution of the algorithm.

- Suppose W connects the output of G_i to the u'th input of G_j, where $u = 1$ or 2. Let C be the last commitment in the selected row of T_i (representing the output bit from G_i) and let X be the u'th commitment in the selected row of T_j. Associate to W the pair C, X.

- Suppose W connects input bit y to input number u of gate G_i ($u = 1$ or 2), and let A be the u'th commitment in the selcted row of T_i. If the commitment Y_W has not been defined yet, let $Y_W = A$. Associate to W the pair Y_W, A.

For all wires, P must now show that the associated pair of commitments contain the same bit. Clearly, this gives at most $2n$ pairs of commitments that must checked for equality. For commitments with unbounded q, or bounded commitments where $\delta l \geq 2n$, P completes these equality proofs by opening only one commitment, by running the Equality protocol shown above. Otherwise, the bits to be compared are distributed over several commitments holding δl bits each, so P will need to open $2n/(\delta l)$ commitments.

PROOF, Step 4
P opens the last commitment in the selected row of T_n (to reveal 1, in order to convince V about the final result of the computation in the circuit).

VERIFICATION OF PROOF
If V rejected any of the proofs in the preprocessing, V rejects immediately. V selects the rows designated by the information from Step 2 of the proof. V computes the pairs of commitments used by P in Step 3, and verifies that P have proved that all pairs contain equal bits (this amounts to verifying that P has correctly opened one or more commitments to reveal 0). Finally V verifies that the commitment opened in Step 4 was correctly opened to reveal 1.

The following is immediate from inspection of the protocol description:

Lemma 2.3 *If Φ is satisfiable and P follows the protocol, then V always accepts. The communication complexity of the protocol is for the preprocessing $12ln + \alpha(n, l, k)$ bits, where $\alpha(n, l, k)$ is the communication complexity for doing all the interactive proofs required in the preprocessing; and for the proof phase $2(n + l)$ or $(2 + 2/\delta)n + l$ bits.*

2.4 An Alternative Approach Based on Span Programs

The contents of this section can be found in Appendix A

2.5 Zero-Knowledge Proof for QBF

In [23], Shamir gave the first proof that $IP = PSPACE$, by exhibiting an interactive proof system for the PSPACE complete QBF problem. A little later, Shen [24], building on Shamirs ideas, gave a somewhat more efficient proof system for QBF, which appears to be the most efficient proof system known for QBF.

In this section, we sketch how our techniques may be applied to transform Shens proof system into a zero-knowledge proof system with the essentially the same communication and round complexity.

By examining Shen's protocol, one finds that all the work done takes place in a finite field $GF(q)$ for some prime q. If, for a QBF instance of length n, we want error probability negligible in n, say 2^{-n}, the analysis of the protocol shows that this can be done by using a q of bit length $O(n)$.

By further inspection of the protocol, one finds that in each round of the protocol, the prover sends the coefficients of some polynomial, the verifier checks this polynomial, and returns a random element in the field. The operations done by the verifier in order to check the polynomials received all fall in one of the following categories:

1. Evaluate a polynomial received from the prover in a point chosen by the verifier, or in a constant point.
2. Add or multiply a constant number of values computed as in 1).
3. Compare values computed as in 1) or 2).
4. The final step: insert all random values chosen by the verifier into a multivariate polynomial efficiently computable from the input QBF instance. Compare the result to a value obtained from the previous rounds.

Our proposed modification of the protocol now simply consists of having the prover communicate his polynomials by in stead sending commitments to each of the coefficients.

By our assumptions on commitments, it is clear that this affects the number of bits needed to send a polynomial by at most a constant factor, and furthermore that the verifier can on his own compute commitments to results of operations of type 1. For the multiplications in 2), the prover supplies a commitment containing the result of each such multiplication. Therefore, at the end of the interaction, the verifier has for each multiplication in the original protocol a set of triples of commitments (A, B, C) containing values (a, b, c), also he has one commitment D together with a value d that can be computed efficiently from the QBF instance. The verifier now only needs to be convinced that for each triple, it holds that $ab \bmod p = c$, and that D contains d. From our assumptions on commitments, it follows directly that the prover can convince the verifier about these facts in honest verifier zero-knowledge. Standard techniques can then be used to build a zero-knowledge protocol.

As can be seen from the following, the multiplication protocol we have is constant round and communicates a constant number of commitments. We therefore get a protocol with the same round and communication complexity, up to a constant factor.

Intuitively, this new protocol is zero-knowledge, because the verifer never sees any of the values chosen by the prover, only computationally useless commitments to them. Soundness is preserved, because the prover must, even in the transformed protocol, decide on the polynomial to send in a given round, before he sees the random field element chosen by the verifier in that round. A more precise statement of our result follows in Section 4.

3 Commitment Schemes Based on Group Homomorphisms

A commitment scheme of the kind we use consists of a function $\texttt{commit} : \{0,1\}^l \times [0..q[\to \{0,1\}^l$, whose description is output by a probabilistic polynomial time *generator* on input 1^l and a prime q, where l is a security parameter. This is done in the *set-up phase* of the commitment scheme. The generator may be able to take an arbitrary prime q as input. This is called a generator with *unbounded* q. Or there may be a constant $\delta > 0$, such that the generator works, only if $q = 2^{\delta l}$. This corresponds to the definition in Section 2.1.

We refer to `commit` as the *public key* of the commitment scheme. To commit to an integer $a \in [0..q[$, one chooses r at random from $\{0,1\}^l$ and computes the commitment C as $C \leftarrow \texttt{commit}(r, a)$. The value r masks a. To open a commitment, r, a are revealed, and the verifier verifies that $\texttt{commit}(r, a) = C$.

For interactive proofs, we will need commitments to be *unconditionally binding.* This means that a is uniquely determined from $\texttt{commit}(r, a)$. Of course we also need the scheme to hide a, but the best we can get in this case is that it is *computationally hiding*: the distributions of commitments to any pair of distinct integers are polynomially indistinguishable.

For interactive arguments, we will use commitment schemes with dual properties: *unconditionally hiding.* This means that the a commitment to a has distribution independent of a. Then, with respect to the binding property, the best we can achieve is that the scheme is *computationally binding.* This means that, given the public key, no probabilistic polynomial time algorithm can compute a commitment and open it in two distinct ways, except with negligible probability.

3.1 Basic Definitions

To show how we build commitment schemes of the kind we need, we start with some notation and definitions:

Definition 3.1 *A* Group Homomorphism Generator $\mathcal{G}$ *is a probabilistic polynomial time algorithm which on input* 1^l *outputs a description of two finite Abelian groups* G, H *and a homomorphism* $f : H \rightarrow G$*. Elements in* G, H *can be represented as* l*-bit strings, and the group operation and inverses in* G *and* H *can be computed in polynomial time. Finally, a uniformly chosen element in* H *can be selected in probabilistic polynomial time.*

Definition 3.2 *A group homomorphism generator* $\mathcal{G}$ *is said to be* one-way *if the following holds for any polynomial size family of circuits* $\{\Delta_i | \ i = 1, 2, ..\}$*: on input* f, y*, where* f *is selected by* $\mathcal{G}$ *on input* 1^l *and* y *is uniformly chosen in* $Im(f)$*, the probability that* Δ_l *outputs* $x \in H$ *such that* $f(x) = y$ *is superpolynomially small (in* l*).*

We will need a further property of the generator, which loosely speaking says that f is as hard to invert in points of form y^i as it is to invert it in y, as long as $0 < i < q$, but inversion is easy in points of form y^q:

Definition 3.3 *A group homomorphism generator* $\mathcal{G}$ *is said to be* q-one-way *if it is one-way, takes a prime* q *as additional input, and there is a polynomial time algorithm satisfying the following: on input* f, z, y, i *where* $0 < i < q$, $y \in G$, $f(z) = y^i$*, it computes* x *such that* $f(x) = y$*. Finally, there is a polynomial time algorithm which on input* y *computes* x' *such that* $f(x') = y^q$*.*

We remark that if f is one-one, and $|H| = q$, q-one-wayness follows trivially from one-wayness.

We are now ready to define the two kinds of generators that will enable us to make the bit commitment schemes we need:

Definition 3.4 *An* unconditionally hiding q-homomorphism generator $\mathcal{G}$ *is a* q*-one-way generator (even though this is the same as the previous definition, we have chosen to give it a separate name, for uniformity with Definition 3.5).*

Definition 3.5 *An* unconditionally binding q-homomorphism generator $\mathcal{G}$ *is a q-one-way generator, which also satisfies that for f generated by $\mathcal{G}$, there exists $y \in G$, such that $yIm(f)$ has order q in the factor group $G/Im(f)$. Furthermore, the distributions $y^i f(r)$ and $y^j f(s)$ for $0 \leq i, j < q$, $i \neq j$ and independently chosen uniform r, s, must be polynomially indistinguishable.*

Informally, what this definition says, is that a y should exist, such that the cosets $yIm(f), y^2Im(f), ..$ are all distinct, and it should be hard to tell the difference between random elements in distinct cosets.

3.2 Commiment Schemes

We are now ready to describe the two types of commitment schemes we have. Throughout, we will assume that a prover P will be generating commitments and sending them to a verifier V. First is an unconditionally hiding scheme:

- **Set-up Phase**: V runs unconditionally hiding q-homomorphism generator $\mathcal{G}$ on input 1^l, to obtain $f : H \to G$. He chooses a random element $y \in Im(f)$, e.g. by choosing an element in H and applying f. Then f, G, H, y are sent to P. V must now give an interactive proof of knowledge that he knows an f-preimage of y. This proof can be easily constructed from the f-preimage protocol in Section 3.3, by using one-bit challenges, and iterating the protocol sequentially.

- **Commitment** to integer $0 \leq a < q$: P chooses random $r \in H$, and sends $\texttt{commit}(r, a) = y^a f(r)$ to V.

- **Opening** commitment C: P sends a, r to V who accepts if and only if $C = \texttt{commit}(r, a)$ and $0 \leq a < q$.

- **Hiding Property**: is clear, since if P has accepted the set-up phase, it follows that (except with exponentially small probability) a commitment will have distribution independent from the value committed to, namely the uniform distribution over $Im(f)$.

- **Binding Property**: If any cheating prover P^* can open a commitment to reveal two different values, he can produce a, r, a', r' such that $a \neq a'$ and $y^a f(r) = y^{a'} f(r')$. Assume without loss of generality that $a > a'$. Then $y^{a-a'} = f(r'r^{-1})$, which means we can find a preimage of y by definition of q-one-wayness. This in turn contradicts the assumption that $\mathcal{G}$ is one-way, if P^* is in polynomial time.

Next, we describe an unconditionally binding scheme:

- **Set-up Phase**: P runs unconditionally binding q-homomorphism generator $\mathcal{G}$ on input 1^l, to obtain $f : H \rightarrow G$. He chooses an element $y \in G$ according to Definition 3.5. Then f, G, H, y are sent to V. For some generators V can verify himself that indeed y has the property requested in Definition 3.5. If this is not the case, P must give a zero-knowledge proof that $y \notin Im(f)$. This can be done by a straightforward modification of the classical quadratic non-residuosity protocol from [17].

- **Commitment** to integer $0 \leq a < q$: P chooses random $r \in H$, and sends $\texttt{commit}(r, a) = y^a f(r)$ to V.

- **Opening** commitment C: P sends a, r to V who accepts if and only if $C = \texttt{commit}(r, a)$ and $0 \leq a < q$.

- **Hiding Property**: follows immediately from the assumption in Definition 3.5.

- **Binding Property**: Definition 3.5 guarantees that if V accepts the set-up phase, commitments to different values will be in distinct cosets of $Im(f)$.

It should be clear from the definition of these commitments that both types have the additive homomorphism property required in our protocols: suppose we are given commitments to values a and b. Let j be such that $a + b = (a + b) \bmod q + jq$, and let t be such that $f(t) = y^{jq}$. Note that by assumption, t is easy to compute. It then holds that $\texttt{commit}(r, a) \cdot \texttt{commit}(s, b) = \texttt{commit}(rst, (a + b) \bmod q)$. In a similar way, it follows that $\texttt{commit}(r, a)^c = \texttt{commit}(r', ca \bmod q)$ and $y^c \cdot \texttt{commit}(r, a) = \texttt{commit}(r'', (c + a) \bmod q)$ for a constant c and easily computable values $r', r'' \in H$.

3.3 Proofs for Bit Commitments and Multiplication

We now turn to the required protocol for showing that a commitment contains a 0/1 value. For this, it turns out to be sufficient to be able to prove knowledge of a preimage under f. We have the following protocol, which can used for any f generated by a q-one-way generator, and is a generalization of Schnorr's discrete log protocol [22]:

f-PREIMAGE PROTOCOL
Input: f and $u \in G$. P knows v, such that $f(v) = u$.

1. P chooses $r \in H$ at random and sends $m = f(r)$ to V.

2. V chooses a random number e, so that $0 \leq e < q$ and sends it to P.

3. P sends $z = rv^e$ to V, who checks that $f(z) = mu^e$.

The properties of this protocol are the following:

Lemma 3.6 *If P, V follow the protocol, V always accepts. From two accepting conversations $(m, e, z), (m, e', z')$, where $e \neq e'$, one can efficiently compute v such that $f(v) = u$. Finally, the protocol is honest verifier zero-knowledge.*

Proof The first claim is trivial. The second follows directly from the definition of q-one-wayness. Finally, conversations with the honest verifier are simulated by choosing at random e, z, computing $m = f(z)u^{-e}$ and outputting (m, e, z). □

It is clear that this protocol can be used to show that a commitment C contains 0, by using $u = C$, and that it contains 1 by using $u = Cy^{-1}$. We may now use the proof of partial knowledge technique from [10] to make a protocol in which P proves that C contains 0 or 1, without revealing which is the case. This involves running the protocol for 0 in parallel with the protocol for 1, but have V issue only one challenge s. Now P must answer challenges e_0, e_1 in the two parallel instances, such that $e_0 + e_1 = s$ (for details, please refer to [10]).

The resulting protocol is referred to as a *bit commitment proof.* It is still honest verifier zero-knowledge, and is a proof of knowledge with error probability $1/q$ that P can open C as 0 or 1. Its communication complexity is $4l + \log q$ bits. Suppose that for some protocol, we need error probability 2^{-k}. For this, we will need to repeat the 0/1 protocol in parallel $\lceil k/\log q \rceil$ times, leading to a communication complexity of $4l\lceil k/\log q \rceil + k$ bits.

The final auxiliary protocol we need is a *multiplication protocol*, an interactive proof that commitments A, B, C contain a, b, c for which $c = ab \bmod q$. Assume P knows how to write the commitments in the form

$$A = y^a f(r),\ B = y^b f(u),\ C = y^{ab \mathrm{mod} q} f(s).$$

Now observe that if we choose j such that $ab = (ab) \bmod q + jq$ and set $t = f^{-1}(y^{-jq})su^{-a}$, then t is easily computable by P, and

$$C = B^a f(t).$$

Conversely, assuming that you can open A and B to reveal a, b, knowledge of such a t implies you can open C to reveal $ab \bmod q$.

This leads to a protocol proving what we want:

MULTIPLICATION PROTOCOL
Input: f and commitments A, B, C. P knows a, r, t, b, u, such that $A = y^a f(r)$, $C = B^a f(t)$ and $B = y^b f(u)$.

The protocol proceeds by executing the following two 3-step protocols in parallel, using the same challenge e in both instances. The first is intended to verify that A, C have the correct form, while the second verifies that the prover can open B:

1. First protocol:
   (a) P chooses $x \in Z_q$ and $s_1, s_2 \in H$ at random and sends $m_1 = y^x f(s_1)$, $m_2 = B^x f(s_2)$ to V.
   (b) V chooses a random number e, so that $0 \leq e < q$ and sends it to P.
   (c) P sets $z = (x + ea) \bmod q$ and chooses i such that $z = x + ea + iq$. He then computes $w_1 = s_1 r^e f^{-1}(y^{-iq})$ and $w_2 = s_2 t^e f^{-1}(B^{-iq})$. He sends z, w_1, w_2 to V, who verifies that $y^z f(w_1) = m_1 A^e$ and $B^z f(w_2) = m_2 C^e$.
2. Second protocol:
   (a) P chooses $d \in Z_q$ and $s \in H$ at random and sends $m = y^d f(s)$ to V.
   (b) V chooses a random number e, so that $0 \leq e < q$ and sends it to P.
   (c) P sets $z = (d + eb) \bmod q$ and chooses j such that $z = d + eb + jq$. He then computes $w = su^e f^{-1}(y^{-jq})$. He sends z, w to V, who verifies that $y^z f(w) = mB^e$

The properties of this protocol are the following:

Lemma 3.7 *If P, V follow the protocol, V always accepts. From two accepting conversations $(m, m_1, m_2, e, z, w, w_1, w_2), (m, m_1, m_2, e', z', w', w_1', w_2')$, where $e \neq e'$, one can efficiently compute a, r, b, u, s such that $A = y^a f(r)$, $B = y^b f(u)$, $C = y^{ab \bmod q} f(s)$. Finally, the protocol is honest verifier perfect zero-knowledge.*

Proof Easy modification of the proof of Lemma 3.6. □

The communication complexity of the multiplication protocol is $6l + 3 \log q$ bits. Suppose that for the main protocol, we need error probability 2^{-k}. For this, we will repeat it in parallel $\lceil k / \log q \rceil$ times.

4 Results for the Main Protocols

In this section we state, without proof, the results we obtain for our main protocols when using the commitment schemes from the previous section. The results are restated and proved in Appendix B.

For formal definitions of proof systems, completeness, soundness and zero-knowledge, please refer to [17]. In the case of arguments, completeness and zero-knowledge are as for proof systems, but for soundness, we treat the error probability in a way similar to the soundness error of proofs of knowledge as defined by Bellare and Goldreich [3]: we will show that if a cheating prover can convince the verifier with probability $\epsilon > 2^{-k}$, then he can break the bit commitment scheme in expected time polynomial in l and $1/(\epsilon - 2^{-k})$.

We remark that all our communication complexity results are computed without including the complexity of setting up the commitment schemes (Step 0 in the protocol descriptions). This is of course motivated by the fact that the same commitment scheme instance can be reused in many protocol executions. However, there are several cases, where including the setup step would make no difference. This is true in general for Theorem 4.3, and for Theorems 4.4, 4.6 when based on the Diffie-Hellman generator described later.

The general strategy for proving the results for our protocols is the following: we first show directly that the main protocols as described earlier are honest verifier zero-knowledge. We cannot get zero-knowledge in general using standard resettable simulation since the prover must in all our protocols answer a challenge consisting of many bits. For arguments, this is solved by having the verifier prove initially knowledge of a trapdoor for the commitment scheme; the simulator can extract the trapdoor, and then simulate easily. For interactive proofs, we use the technique of having the verifier commit to his challenge in advance. This allows simulation as shown by Goldreich and Kahan [14]

4.1 Results for Non-Interactive SAT Protocols with Preprocessing

Lemma 4.1 *The protocol in Subsection 2.3 using commitments constructed from an unconditionally hiding q-homomorphism generator with unbounded q is a perfect honest verifier zero-knowledge argument with preprocessing for Boolean Circuit Satisfiability. The communication complexity of the preprocessing is $O(nl+k)$ bits, while the proof phase has size $O(n+l)$. If the generator has bounded q, the conclusion is the same, except that the communication complexity of the prepro-*

cessing is $O(n)max(k,l)$ bits.

Before we give the corresponding result for unconditionally binding generators, we note that an unconditionally binding generator cannot have unbounded q, because it leads to l-bit commitments from which the contents is uniquely determined, and so we must have at least that $q < 2^l$.

Lemma 4.2 *The protocol in Subsection 2.3 using commitments constructed from an unconditionally binding q-homomorphism generator (with bounded q) is a computational honest verifier zero-knowledge proof with preprocessing for Boolean Circuit Satisfiability. Communication complexity of the preprocessing is $O(n)max(k,l)$ bits, while the proof phase has size $O(n+l)$.*

It now only remains to modify these protocols to be zero-knowledge in general, of course without loosing efficiency. We obtain the following:

Theorem 4.3 *If there exists an unconditionally hiding q-homomorphism generator with unbounded q then there exists a non-interactive perfect zero-knowledge argument with preprocessing for Boolean Formula Satisfiability. The communication complexity of the preprocessing is $O(nl+k)$ bits, while the proof phase has size $O(n+l)$. If the generator has bounded q, the conclusion is the same, but the communication complexity of the preprocessing becomes $O(n)max(k,l)$ bits.*

Theorem 4.4 *If there exists an unconditionally binding q-homomorphism generator (with bounded q) then there exists a non-interactive zero-knowledge proof with preprocessing for Boolean Formula Satisfiability, such that the communication complexity of the preprocessing is $O(n)max(k,l)$ bits, while the proof phase has size $O(n+l)$.*

4.2 Results for Arithmetic Circuit Protocols

Recall that the protocols in Section 2.2 were defined for an n-bit prime q, error probability 2^{-k}, and a circuit with m inputs and t multiplication gates.

Lemma 4.5 *The protocol in Subsection 2.2 using commitments constructed from an unconditionally hiding q-homomorphism generator is a perfect honest verifier zero-knowledge argument for the arithmetic circuit problem. When using commitments constructed from an unconditionally binding q-homomorphism generator we obtain an honest verifier computationally zero-knowledge proof. The communication complexity is $O((m+t)(l+n)\lceil k/n \rceil)$ bits in either case.*

Theorem 4.6 *If there exists an unconditionally hiding, resp. an unconditionally binding q-homomorphism generator then there exists a perfect zero-knowledge argument, resp. a computational zero-knowledge proof for the arithmetic circuit problem. The communication complexity is $O((m+t)(l+n)\lceil k/n \rceil)$ bits in either case.*

4.3 Result for the Zero-Knowledge QBF Protocol

Theorem 4.7 *If there exists an unconditionally binding q-homomorphism generator (with bounded q), then there exists a zero-knowledge interactive proof system for the QBF problem with the same asymptotic round and communication complexity as Shen's interactive proof system when designed to have error probability 2^{-n} for a length n QBF instance.*

Proof sketch
The zero-knowledge protocol described in Subsection 2.5 consists of first a stage where the prover and verifier go through "the same" interaction as in the original proof system, except that the prover sends commitments to his messages. Then a stage, where the prover convinces the verifier that a set of relations hold between the committed values. This stage is only honest verifier zero-knowledge as described in Section 2.5, but can be made zero-knowledge with no essential loss of efficiency in the same way as in the proof of Theorem 4.4, using the method from [14].

Having said this, the proof that our modified protocol is a zero-knowledge proof system for QBF is a straightforward modification of the proof from [6] that everything in IP has a zero-knowledge proof system if one-way functions exist. Specifically, note the following: Like ours, the protocol built in [6] is a modification of an Arthur-Merlin interactive proof system with one-sided error (the honest prover always convinces the verifier). The transformation from [6] results in a two-stage protocol of the same form as ours. And finally, [6] assumes that the prover encrypts his messages using polynomially secure probabilistic encryption. This corresponds to the hiding property of our commitments. □

5 Examples of Group Homomorphisms

Recall that any of our generators have 1^l and a prime q as parameters. Generators with bounded q include as part of their definition a constant δ.

5.1 RSA based One-Way Homomorphisms

We first show an example of an unconditionally hiding homomorphism generator based on RSA:

RSA GENERATOR
The generator selects an RSA modulus $N = p_1 p_2$ of bit length l, for primes p_1, p_2, such that $(q, (p_1 - 1)(p_2 - 1)) = 1$. The output is N.

For this generator, we define $H = G = Z_N^*$, and $f(x) = x^q \bmod N$.

Lemma 5.1 *Under the RSA assumption, the RSA generator is an unconditionally hiding q-homomorphism generator, with unbounded q.*

Proof If $q > N$, it must be prime to $\phi(N)$, whence f is surjective. Hence deciding membership of some y in $Im(f)$ only consists of verifying the q is a prime and that $(y, N) = 1$. Otherwise, a zero-knowledge proof must be provided that y is a qth power modulo N.

The generator is clearly one-way under the usual RSA assumption.

The only other requirement that is not completely trivial is q-one-wayness: assume we have z, y, i such that $y^i = z^q \bmod N$. Since $1 \leq i < q$, i is prime to q, so take α, β such that $\alpha i + \beta q = 1$. We claim that $x = z^\alpha \cdot y^\beta$ is the preimage of y we are looking for:

$$f(x) = z^{\alpha q} \cdot y^{\beta q} = y^{\alpha i} \cot y^{\beta q} = y \bmod N$$

□

One can also base an unconditionally binding generator on an RSA-like function. The resulting commitment/encryption scheme was first discovered by Benaloh [7] in the context of verifiable secret sharing.

q-RESIDUOSITY GENERATOR
The generator selects an RSA modulus $N = p_1 p_2$ of bit length l, for primes p_1, p_2, subject to $q | (p_1 - 1)(p_2 - 1)$ and $\delta = \log q / \log N$. The output is N.

For this generator, we define $H = G = Z_N^*$, and $f(x) = x^q \bmod N$.

By the *q'th residusity assumption*, we mean the assumption that random elements in distinct cosets of $Im(f)$ as defined here are polynomially indistinguishable. This is a natural generalization of the well known quadratic residuosity assumption.

Lemma 5.2 *Under the q'th residuosity assumption, the q-residuosity generator is an unconditionally binding q-homomorphism generator.*

Proof The proof of q-one-wayness is the same as for the RSA generator (note that our assumption in particular implies that f is one-way). The element y that is required to exist in Definition 3.5 can be chosen as any element not in $Im(f)$. □

5.2 Diffie-Hellman and Discrete Log Based One-Way Homomorphisms

We first give a generator based on the discrete log problem modulo a prime number. The commitment scheme resulting from this generator was first discovered by Pedersen [21] in the context of verifiable secret sharing.

DISCRETE LOG GENERATOR
The generator selects randomly a prime p of bit length l, subject to $\delta = \log q / \log p$ and $q | p-1$, where $0 < \delta < 1$ is a constant. It also selects $g \in Z_p^*$, such that g generates the (unique) subgroup in Z_p^* of order q. The output is p, g.

For this generator, we define $H = Z_q, G =< g >$, and $f(x) = g^x \bmod p$. When using this generator as basis for our protocols, we will assume that a party receiving an element u supposedly in G always verifies that $u^q = 1$ and stops the protocol if not.

Lemma 5.3 *Assume that any probabilistic polynomial time algorithm solves the discrete log problem modulo prime numbers as selected by the Discrete Log Generator with superpolynomially small probability. Then the Discrete Log Generator is an unconditionally hiding q-homomorphism generator with bounded q.*

Proof It is clear that deciding membership of y in $Im(f)$ amounts to verifying that p, q are primes, that $q | p-1$ and that $g^q = y^q = 1 \bmod p$. Inverting f is exactly the discrete log problem, which we assumed hard. Finally, for q-one-wayness, from $y^i = g^z \bmod p$, we obtain

$$y = g^{z \cdot i^{-1} \mathrm{mod} q} \bmod p = f(z \cdot i^{-1} \bmod q)$$

which is possible since $i < q$ and so prime to q. □

We remark that nothing prevents us from using other groups of prime order, such as for example the group on an appropriately chosen elliptic curve.

Finally, we show an example of an unconditionally binding generator, based on the Diffie-Hellman problem [11]:

DIFFIE-HELLMAN GENERATOR
The generator selects randomly a prime p of bit length $l/2$, subject to $\delta = \log q/l$ and $q|p-1$, where $0 < \delta < 1/2$ is a constant. It also selects $g \in Z_p^*$, such that g generates the (unique) subgroup in Z_p^* of order q, and finally a random $h \in < g >$. The output is p, g, h.

For this generator, we define $H = Z_q, G =< g > \times < g >$, and $f(x) = (g^x \bmod p, h^x \bmod p)$ [5].

Recall that (p, q, g, h) can be used as a public key to encrypt an element $m \in < g >$ by choosing r at random and letting the ciphertext be $(g^r \bmod p, mh^r \bmod p)$ [12]. We will call this Diffie-Hellman encryption. Recall also the notion of polynomial security, defined by Goldwasser and Micali [15], which says that random encryptions of distinct messages are poynomially indistinguishable.

Lemma 5.4 *If Diffie-Hellman encryption is polynomially secure, then the Diffie-Hellman generator is an unconditionally binding q-homomorphism generator.*

Proof Considering Diffie-Hellman encryption in group theoretic terms, we are in fact choosing a random representative of the coset $(1, m) \cdot Im(f)$ in G. Hence polynomial security is equivalent to saying that random elements in cosets of form $(1, m^i)Im(f)$, $(1, m^j)Im(f)$ are polynomially indistinguishable. So for the requirement in Definition 3.5, we can use any $y = (1, m)$, where $m \neq 1 \bmod p$.

Clearly, Diffie-Hellman cannot be polynomially secure, unless $x \to g^x \bmod p$ is hard to invert, and f is easily seen to be as hard to invert as this mapping. q-one-wayness follows in the same way as for the discrete log generator. □

References

[1] D. Beaver: *Efficient Multiparty Protocols Using Circuit Randomization*, Proceedings of Crypto 91, Springer-Verlag LNCS, 1992, pp. 420–432.

[2] L. Babai, L. Fortnow, L. Levin and M. Szegedi: *Checking Computations in Poly-logarithmic Time*, Proceedings of STOC '91.

[3] M. Bellare and and O. Goldreich: *On Defining Proofs of Knowledge*, Proceedings of Crypto '92, Springer Verlag LNCS, vol. 740, pp. 390–420.

[5] The remark on verification of memebership in G for the Discrete Log Generator also applies here

[4] J. Boyar, G. Brassard and R. Peralta: *Subquadratic Zero-Knowledge*, Journal of the ACM, November 1995.

[5] G. Brassard, D. Chaum and C. Crépeau: *Minimum Disclosure Proofs of Knowledge*, JCSS, vol.37, pp. 156–189, 1988.

[6] M.Ben-Or, O.Goldreich, S.Goldwasser, J.Håstad, J.Kilian, S.Micali and P.Rogaway: *Everything Provable is Provable in Zero-Knowledge*, Proceedings of Crypto 88, Springer Verlag LNCS series, 37–56.

[7] J. Benaloh: *Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret*, Proc. of Crypto 86, Springer Verlag LNCS series, 251–260.

[8] J. Benaloh and J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto 88, Springer Verlag LNCS series, 25–35.

[9] R. Cramer and I. Damgård: *Linear Zero-Knowledge*, Proc. of STOC 97.

[10] R. Cramer, I. Damgård and B. Schoenmakers: *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, Proceedings of Crypto '94, Springer verlag LNCS, vol. 839, pp. 174–187.

[11] W. Diffie and M. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory IT-22 (6): 644–654, 1976.

[12] T. ElGamal, *A Public-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms*, IEEE Transactions on Information Theory, IT-31 (4): 469–472, 1985.

[13] L.Fortnow: *The complexity of Perfect Zero-Knowledge*, Adv. in Computing Research, vol.5, 1989, 327–344.

[14] O. Goldreich and A. Kahan: *How to Construct Constant-Round Zero-Knowledge Proof Systems for NP*, Journal of Cryptology, (1996) 9: 167–189.

[15] S. Goldwasser and S. Micali: *Probabilistic Encryption*, JCSS, vol.28, 1984.

[16] O. Goldreich, S. Micali and A. Wigderson: *Proofs that yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design*, Proceedings of FOCS '86, pp. 174–187.

[17] S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, Vol. 18, pp. 186-208, 1989.

[18] J. Kilian: *A note on Efficient Proofs and Arguments*, Proceedings of STOC '92.

[19] J. Kilian: *Efficient Interactive Arguments*, Proceedings of Crypto '95, Springer Verlag LNCS, vol. 963, pp. 311–324.

[20] M. Karchmer and A. Wigderson: *On Span Programs*, Proc. of Structure in Complexity, 1993.

[21] T. Pedersen: *Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing*, proc. of Crypto 91, Springer Verlag LNCS, vol. 576, pp. 129–140.

[22] C. P. Schnorr: *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, 4 (3): 161–174, 1991.

[23] A.Shamir: *IP=PSPACE*, Journal of the ACM, vol.39 (1992), 869-877.

[24] A. Shen: *IP=PSPACE, Simplified Proof*, Journal of the ACM, vol.39 (1992), pp.878-880.

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

A An Alternative Approach Based on Span Programs

The span program is a computational model based on linear algebra over (finite) fields, introduced by Karchmer and Wigderson [20].

First, it is immediately clear that our protocol for general arithmetic circuits applies to show satisfiability of a span program. Since most of the computation required in a span program is linear, the resulting protocol would have complexity depending only linearly on the size of the span program. Since span programs may be more powerful for some problems than Boolean circuits, this can for such problems lead to more efficient protocols than the ones obtained by going through a reduction to SAT.

Moreover, the span approach leads to particularly efficent protocols when applied to the SAT problem: based on span programs and our commitment schemes from Section 2.1 it is possible to construct non-interactive zero knowledge proofs for SAT with preprocessing, achieving the same *asymptotic* communication complexities as our protocol from Section 2.3. Although the constants involved are less favorable, we outline this approach based on span programs, since we believe that future improvements on the communication complexities might very well proceed along these lines. Let C be the Boolean circuit to be proved satisfiable. Our goal is to find an efficient span program M_C, i.e. one with a "small" number of rows and columns, such that the Boolean function computed by it is satisfiable if and only if C is [6] (please refer to [20] for definitions concerning span programs).

To this end, it is sufficient to pass first to a Boolean formula Φ_C that is satisfiable if and only if C is: it is quite straightforward to construct a span-program computing the same function as a given Boolean formula (inductive argument). Moreover, this construction yields a span program with at most $O(m)$ rows and columns, where m is the size of the formula.

To faciliate construction of our protocol based on span programs and our commitment schemes, some further properties are very useful.

If we pass from C to Phi by taking it as the conjunction over the $|C|$ formulas that check the computation of C at each gate, and derive the span program from this particular formula Φ, it turns out that the matrix that defines the span program can be chosen as a $0/1$-matrix and that the coefficients in a linear combination of the rows leading to the root of the span program can be selected from the set $\{-1, 0, 1\}$, regardless the underlying field we have chosen for the span program. Finally, we note that the columns of the span program thus obtained have constant Hamming weight.

With M_C constructed from C as outlined above, to show that C is satisfiable it is sufficient to show that there exists a suitable linear combination of the rows leading to the root of the span program. There is a direct correspondence between a satisfying assignment of C and the coefficients of that linear combination.

Without giving any further details here, we state that even if those coefficients are committed to by means of a commitment scheme as defined in Section 2.1, it is still possible to perform the necessary linear algebraic operations on the coefficients hidden in the commitments and to show, non-interactively and in zero knowledge, the satisfiability of the span program and hence that of C. The pre-processing phase is similar to the one from Section 2.3.

We finally note that finding an even more efficient span program M_C that is satisfiable if and only if C, would probably yield even more efficient non-interactive zero knowledge proofs for SAT with preprocessing.

[6] Note that we do not require that C and $M(C)$ compute the same Boolean function

B Results With Proofs for the Main Protocols

In this section we restate and prove the results we obtain for our main protocols when using the commitment schemes we have presented.

For formal definitions of proof systems, completeness, soundness and zero-knowledge, please refer to [17]. In the case of arguments, completeness and zero-knowledge are as for proof systems, but for soundness, we treat the error probability in a way similar to the soundness error of proofs of knowledge as defined by Bellare and Goldreich [3]: we will show that if a cheating prover can convince the verifier with probability $\epsilon > 2^{-k}$, then he can break the bit commitment scheme in expected time polynomial in l and $1/(\epsilon - 2^{-k})$.

We remark that all our communication complexity results are computed without including the complexity of setting up the commitment schemes (Step 0 in the protocol descriptions). This is of course motivated by the fact that the same commitment scheme instance can be reused in many protocol executions. However, there are several cases, where including the setup step would make no difference. This is true in general for Theorem B.3, and for Theorems B.4, B.6 when based on the Diffie-Hellman generator.

B.1 Results for Non-Interactive SAT Protocols with Preprocessing

Lemma B.1 *The protocol in Subsection 2.3 using commitments constructed from an unconditionally hiding q-homomorphism generator with unbounded q is a perfect honest verifier zero-knowledge argument with preprocessing for Boolean Circuit Satisfiability. The communication complexity of the preprocessing is $O(nl+k)$ bits, while the proof phase has size $O(n+l)$. If the generator has bounded q, the conclusion is the same, except that the communication complexity of the preprocessing is $O(n)max(k,l)$ bits.*

Proof First recall that completeness and communcation complexity was established already in Lemma 2.3, except for the complexity $\alpha(n,k,l)$ of doing the bit commitment proofs in the preprocessing. For the unbounded q case, we will choose $q = max(2^k, 2^{2n})$. For the bounded q case, we have $q = 2^{\delta l}$ for a constant $\delta > 0$. In both cases the communication complexities now follow directly from the remarks on the f-preimage protocol, if one also observes that all the required bit commitment proofs can be done in parallel, using the same k-bit challenge for all of them.

For soundness, assume that some polynomial time P^* can get V to accept a circuit Φ with probability $\epsilon > 2^{-k}$. We will build from this an algorithm Alg which in expected time polynomial in $1/(\epsilon - 2^{-k})$ and l finds either a satisfying assignment, or breaks a given instance of the commitment scheme. This is clearly enough for soundness as defined above.

Alg starts by sending the public key of the bit commitment scheme to P^*. It then simulates the proof that $y \in Im(f)$, with P^* acting as the verifier. Then Alg issues random challenges for the bit commitment proofs, rewinding P^* after each challenge has been answered. When correct answers to two different challenges have been obtained, by Lemma 3.6, we will know how to open all commitments issued. We may assume that all NAND-tables produced are correct, since otherwise the proofs of correctness will give a way to open at least one commitment in a new way.

The proof in Step 3 establishes equality of bits in a set of pairs of commitments selected from the T_i's by multiplying together two sets of commitments and opening the quotient as 0. Given the group element revealed in this opening, if one can open one product, one can compute a way to open the other one to reveal the same value. But since we already know how to open all commitments in the T_i's, we know *a priori* how to open both products, and hence if they do not in fact contain the same value, we can break the commitment scheme.

It now follows that the rows selected in the T_i's represent a consistent computation in the circuit, and since the output is 1, by the opening in Step 4 of the proof, we have a satisfying assignment.

Finally, honest verifier zero-knowledge is easy, since the simulator can use the proof of knowledge given by V in the setup phase to extract a preimage under f of y. This will always succeed if V is honest. Given such a preimage, the simulator can open any commitment any way it wants, and the simulation becomes trivial.

□

Before we give the corresponding result for unconditionally binding generators, we note that an unconditionally binding generator cannot have unbounded q, because it leads to l-bit commitments from which the contents is uniquely determined, and so we must have at least that $q < 2^l$.

Lemma B.2 *The protocol in Subsection 2.3 using commitments constructed from an unconditionally binding q-homomorphism generator (with bounded q) is a computational honest verifier zero-knowledge proof with preprocessing for Boolean Circuit Satisfiability. Communication complexity of the preprocessing is $O(n)max(k, l)$ bits, while the proof phase has size $O(n + l)$.*

Proof The proof of soundness is essentially the same as for the previous lemma. The only difference is that it is enough to observe that if $\epsilon > 2^{-k}$, there *exists* good answers to two different challenges, and so all commitments must contain 0/1 values. Furthermore, the cases where a commitment can be opened in different ways simply cannot occur, and so we always find a satisfying assignment.

We then consider honest verifier zero-knowledge: the simulator executes the setup phase for the commitment scheme according to the protocol. It then constructs all the NAND tables for the preprocessing according to the protocol, except that T_n is constructed such that its third column contains commitments to 1's only. The proofs of correctness of AND-tables are simulated by invoking the honest verifier simulators for the bit-commitment proof and the multiplication protocol a sufficient number of times.

The simulator now chooses an arbitrary set of input bits for the circuit, and does Steps 2 and 3 of the proof according to the protocol. Step 4 is also executed according to the protocol, which is easy by construction of T_n.

The bit commitment proof simulation is sometimes invoked on commitments that are not bit commitments (when we "prove" correctness of the NAND-table used for T_n). But by the hiding property, these commitments have distributions that are polynomially indistinguishable from good values. It follows that the honest verifier simulators we call as subrutines will produce conversations with distribution that is polynomially indistinguishable from the one produced on inputs that are bit commitments.

Finally, the numbers revealed to open various products of commitments have exactly the correct distribution, because this distribution depends only on the form of the expressions defined in the protocol, not on the values in the commitments. □

It now only remains to modify these protocols to be zero-knowledge in general, of course without loosing efficiency. We obtain the following:

Theorem B.3 *If there exists an unconditionally hiding q-homomorphism generator with unbounded q then there exists a non-interactive perfect zero-knowledge argument with preprocessing for Boolean Formula Satisfiability. The communication complexity of the preprocessing is $O(nl + k)$ bits, while the proof phase has size $O(n + l)$. If the generator has bounded q, the conclusion is the same, but the communication complexity of the preprocessing becomes $O(n) max(k, l)$ bits.*

Proof It turns out that the protocol guaranteed by Lemma B.1 can be used here without modification, we only have to require that the proof of knowledge of an

f-preimage of y given by V in the setup phase is iterated $2n$ times so that its error probability is 2^{-2n} (recall that we are using the f-preimage protocol with a 1-bit challenge in each iteration).

With this requirement, we can simulate against an arbitrary V^* as follows: we first execute the setup phase according to the protocol. If V^* gives incorrect answers, we stop and quit (as P would have done). Otherwise, we check if V^* could in any iteration answer also the question we did not ask him in the first run. This is easy to do with rewinding. If yes, we compute a preimage of y, and can trivially simulate the rest of the protocol. If no, we do an exhaustive search for a satisfying assignment (we are only required to simulate if one exists) and do the rest of the protocol by following the prover's algorithm. This clearly produces a correct output distribution. The expected running time is polynomial, since the probability with which we do the exhaustive search is at most 2^{-2n}, and in the search, we need to check at most 2^{2n} possible assigments: since we have at most n binary gates, there can be no more than $2n$ different input bits.

This establishes zero-knowledge. Soundness and asymptotic communication complexity are the same as in Lemma B.1. □

Theorem B.4 *If there exists an unconditionally binding q-homomorphism generator (with bounded q) then there exists a non-interactive zero-knowledge proof with preprocessing for Boolean Formula Satisfiability, such that the communication complexity of the preprocessing is $O(n)max(k, l)$ bits, while the proof phase has size $O(n + l)$.*

Proof We will use the protocol guaranteed by Lemma B.2 together with the technique of Goldreich and Kahan [14], where the verifier commits to his challenge before the prover sends the first message. If an unconditionally hiding commitment scheme is used, P gets no information on V's challenge ahead of time, and hence the soundness is not affected. On the other hand, simulation becomes possible, because the simulator can first get the verifier to open his commitment, and then invoke the honest verifier simulator using the fact that the challenge is now known ahead of time. For a solution of the subtle technical problems with this, see [14].

To establish the required commitment scheme (with V as the committer), notice that we already assume that we have a q-one-way generator, so in the set-up phase, the prover can, once and for all, publish a $y' \in Im(f)$ (in addition to the $y \notin Im(f)$). By the way we choose q in Lemma B.2, this allows V to commit to any l-bit value. If k is larger than this, more commitments must be used. An easy calculation shows that this adds $O(max(k, l))$ bits to the complexity, so the asymptotic behavior stays the same as in Lemma B.2. □

B.2 Results for Arithmetic Circuit Protocols

Recall that the protocols in Section 2.2 were defined for an n-bit prime q, error probability 2^{-k}, and a circuit with m inputs and t multiplication gates.

Lemma B.5 *The protocol in Subsection 2.2 using commitments constructed from an unconditionally hiding q-homomorphism generator is a perfect honest verifier zero-knowledge argument for the arithmetic circuit problem. When using commitments constructed from an unconditionally binding q-homomorphism generator we obtain an honest verifier computationally zero-knowledge proof. The communication complexity is $O((m+t)(l+n)\lceil k/n \rceil)$ bits in either case.*

Proof The communication complexity follows from Lemma 2.1 and the remarks following Lemma 3.7.

We then first handle the case of using an unconditionally hiding generator:

For soundness, assume that some polynomial time P^* can get V to accept a circuit Ψ and output value y with probability $\epsilon > 2^{-k}$. We will build from this an algorithm Alg which in expected time polynomial in $1/(\epsilon - 2^{-k})$ and l finds either a assignment to the inputs that lead to output y, or breaks the commitment scheme. This is clearly enough for soundness as defined above.

Alg starts by sending the public key of the bit commitment scheme to P^*. It then a zero-knowledge proof that $y \in Im(f)$, with P^* acting as the verifier. Then Alg issues random challenges for the multiplication proofs and the proofs of knowledge of contents for $I_1, .., I_m, T_1, ..., T_t$, rewinding P^* after each challenge has been answered. When correct answers to two different challenges have been obtained, by Lemma 3.7, we know how to open all involved commitments. If the input values in $I_1, ..., I_m$ that we know lead to input values to a multiplication gate that are not consistent with the values we know from the multiplication protocol, we can break the commitment scheme. If not, the input values we have found do in fact lead to y as output from Ψ.

Honest verifier zero-knowledge is easy, since the simulator can use the proof of knowledge given by V in the setup phase to extract a preimage under f of y. This will always succeed if V is honest. Given such a preimage, the simulator can open any commitment any way it wants, and the simulation becomes trivial.

Now to the case of using an unconditionally binding generator:

Soundness can be argued in the same way as above, except that it is enough to observe that if $\epsilon > 2^{-k}$, there *exists* good answers to two different challenges, and so all multiplication gates must have correctly related input/output values assigned. Furthermore, the cases where a commitment can be opened in different ways simply cannot occur, and so we always find a good set of input values.

For honest verifier zero-knowledge, note that Ψ is a natural way defines a linear mapping from $GF(q)^{m+t}$ to $GF(q)$, where one starts by the inputs to Ψ plus the outputs from all multiplications and use the linear operations in Ψ to compute the output. If the provers claim that Ψ may produce y is true, then y must be in the image of this mapping. It follows that the simulator can, by solving a linear system of equations, compute values to put in the $I_1, ..., I_m, T_1, ..., T_t$ that will lead to a commitment containing y when the verifier computes the commitment representing the output value. So the simulator constructs the commitments this way. All that remains now is to invoke the honest verifier simulators for multiplication proofs and proofs of contents an appropriate number of times, and open the output commitment to reveal y.

For correctness of the output distribution, note that values in commitments assigned to multiplication gates may not have the multiplicative relation they always obey in real conversations. But by the hiding property, the commitments to these values are polynomially indistinguishable from commitments to correct values. It follows that the (polynomial-time) honest verifier simulator we call as subrutine, produces from such commitments conversations that are polynomially indistinguishable from real conversations. Hence we get computational zero-knowledge. □

Theorem B.6 *If there exists an unconditionally hiding, resp. an unconditionally binding q-homomorphism generator then there exists a perfect zero-knowledge argument, resp. a computational zero-knowledge proof for the arithmetic circuit problem. The communication complexity is $O((m+t)(l+n)\lceil k/n \rceil)$ bits in either case.*

Proof We can use the protocols guaranteed by Lemma B.5 and transform them to zero-knowlegde protocols using exactly the same methods as in the proofs of Theorems 4.3 and 4.4. □

B.3 Result for the Zero-Knowledge QBF Protocol

Theorem B.7 *If there exists an unconditionally binding q-homomorphism generator (with bounded q), then there exists a zero-knowledge interactive proof sys-*

tem for the QBF problem with the same asymptotic round and communication complexity as Shen's interactive proof system when designed to have error probability 2^{-n} *for a length* n *QBF instance.*

Proof sketch

The zero-knowledge protocol described in Subsection 2.5 consists of first a stage where the prover and verifier go through "the same" interaction as in the original proof system, except that the prover sends commitments to his messages. Then a stage, where the prover convinces the verifier that a set of relations hold between the committed values. This stage is only honest verifier zero-knowledge as described in Section 2.5, but can be made zero-knowledge with no essential loss of efficiency in the same way as in the proof of Theorem 4.4, using the method from [14].

Having said this, the proof that our modified protocol is a zero-knowledge proof system for QBF is a straightforward modification of the proof from [6] that everything in IP has a zero-knowledge proof system if one-way functions exist. Specifically, note the following: Like ours, the protocol built in [6] is a modification of an Arthur-Merlin interactive proof system with one-sided error (the honest prover always convinces the verifier). The transformation from [6] results in a two-stage protocol of the same form as ours. And finally, [6] assumes that the prover encrypts his messages using polynomially secure probabilistic encryption. This corresponds to the hiding property of our commitments. □

Recent BRICS Report Series Publications

RS-97-27 **Ronald Cramer and Ivan B. Damgård.** ***Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?*** **November 1997. 33 pp.**

RS-97-26 **Luca Aceto and Anna Ingólfsdóttir.** ***A Characterization of Finitary Bisimulation*****. October 1997. 9 pp. To appear in** ***Information Processing Letters*****.**

RS-97-25 **David A. Mix Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum.** ***Searching Constant Width Mazes Captures the*** AC^0 ***Hierarchy*****. September 1997. 20 pp. To appear in** ***STACS '98: 15th Annual Symposium on Theoretical Aspects of Computer Science Proceedings*****, LNCS, 1998.**

RS-97-24 **Søren B. Lassen.** ***Relational Reasoning about Contexts*****. September 1997. 45 pp. To appear as a chapter in the book** ***Higher Order Operational Techniques in Semantics*****, eds. Andrew D. Gordon and Andrew M. Pitts, Cambridge University Press.**

RS-97-23 **Ulrich Kohlenbach.** ***On the Arithmetical Content of Restricted Forms of Comprehension, Choice and General Uniform Boundedness*****. August 1997. 35 pp.**

RS-97-22 **Carsten Butz.** ***Syntax and Semantics of the logic*** $\mathcal{L}^{\lambda}_{\omega\omega}$**. July 1997. 14 pp.**

RS-97-21 **Steve Awodey and Carsten Butz.** ***Topological Completeness for Higher-Order Logic*****. July 1997. 19 pp.**

RS-97-20 **Carsten Butz and Peter T. Johnstone.** ***Classifying Toposes for First Order Theories*****. July 1997. 34 pp.**

RS-97-19 **Andrew D. Gordon, Paul D. Hankin, and Søren B. Lassen.** ***Compilation and Equivalence of Imperative Objects*****. July 1997. iv+64 pp. Appears also as Technical Report 429, University of Cambridge Computer Laboratory, June 1997. To appear in** ***Foundations of Software Technology and Theoretical Computer Science: 17th Conference*****, FCT&TCS '97 Proceedings, LNCS, 1997.**

RS-97-18 **Robert Pollack.** ***How to Believe a Machine-Checked Proof*****. July 1997. 18 pp. To appear as a chapter in the book** ***Twenty Five Years of Constructive Type Theory*****, eds. Smith and Sambin, Oxford University Press.**