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

Ronald Cramer[1] Ivan Damgård[2]

Abstract. We present a general method for constructing commitment schemes based on existence of q-one way group homomorphisms, in which elements in a finite prime field $GF(q)$ can be committed to. A receiver of commitments can non-interactively check whether committed values satisfy linear equations. Multiplicative relations can be verified interactively with exponentially small error, while communicating only a constant number of commitments. Particular assumptions sufficient for our commitment schemes include: the RSA assumption, hardness of discrete log in a prime order group, and polynomial security of Diffie-Hellman encryption.
Based on these commitments, we give efficient zero-knowledge proofs and arguments for arithmetic circuits over finite prime fields, namely given such 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 m-bit prime, a circuit of size $O(m)$, and error probability 2^{-m}, our protocols require communication of $O(m^2)$ bits. We then look at the Boolean Circuit Satisfiability problem and give non-interactive zero-knowledge proofs and arguments with preprocessing. In the proof stage, the prover can prove any circuit of size n he wants by sending only one message of size $O(n)$ bits. As a final 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.

1 Introduction

In this paper, we present a general method for building commitment schemes, which are based on existence of any family of one-way group homomorphisms with a particular extra property (detailed below). We call such functions *q-one way group homomorphisms*.

Informally speaking, these schemes allow a *prover* to compute a commitment to an element a in the finite prime field $GF(q)$, having sent this commitment to a *verifier*, the prover cannot change his mind about a, still the verifier cannot guess a from the commitment.

Our commitments are small (i.e. if q is an m bit prime, commitments will be of size $O(m)$ bits) and have useful homorphic properties: given any linear equation over $GF(q)$, the verifier can check whether a set of committed values satisfy the equation without communicating with the prover. We give an efficient protocol allowing the prover to convince the verifier that committed values a, b, c satisfy $ab = c$ without revealing anything else about a, b, c. By efficient, we mean that the protocol achieves error probability exponentially small in m, but requires

[1] ETH Zürich
[2] Aarhus University, BRICS (Basic Research in Computer Science, center of the Danish National Research Foundation)

only communication of a constant number of commitments. Other auxiliary protocols allow the prover to convince the verifier that a commitment contains 0 or 1; and to convince him that pairs of committed *bits* $(c_1, d_1), ..., (c_m, d_m)$ satisfy $c_i = d_i, i = 1..m$ by opening only one commitment.

We give examples of concrete assumptions sufficient for the existence of q-one way homomorphisms, including the RSA assumption, hardness of discrete log in a prime order group, and polynomial security of Diffie-Hellman encryption. When instantiating our commitments using these concrete assumptions, we get some examples of commitment schemes that were known, while others are new. However, no efficient multiplication protocol were known for any of these schemes before. We consider this multiplication protocol and our unified view of many apparently different commitment schemes to be an important technical contributions of this paper [3]. In recent work by Gennaro et al. [21] and Cramer et al. [9], our commitment schemes have been used as an essential tool to build efficient multiparty computations protocols.

Perhaps the most obvious application of commitment schemes in general is for building Zero-Knowledge interactive proofs [20] and arguments [5]. These are protocols allowing a prover to convince a verifier that a statement 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, [15]). If one-way functions exist, then all languages in IP (and hence in NP) have zero-knowledge proofs [19, 6]. 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 [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, 22, 23, 8]. 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? We do not have a general or final answer to this (hence the question mark in the title). But we do show that our commitment schemes can be applied to build protocols implying that in some cases, zero-knowledge may indeed be almost or entirely for free, as far as communication complexity is concerned.

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* (ACP). For a field

[3] In [16], a commitment scheme is given, together with a multiplication protocol with properties somewhat similar to ours. That protocol, however, only works under a specialized strong version of the RSA assumption, and can only be used to make statistical zero-knowledge arguments (as opposed to perfect zero-knowledge arguments as well as zero-knowledge proofs in our case).

$GF(q)$, where q is an m-bit prime, a circuit of size $O(m)$, cryptographic security parameter m and error probability 2^{-m}, our protocols require communication of $O(m^2)$ bits. A more detailed account of the performance of our protocol is given in Theorem 5.3 and shows that the circuit actually only influences the complexity through the number of inputs and multiplications - linear operations are for free. If the circuit involves m multiplications, the best previously known method is to rewrite the multiplications to Boolean circuits and use the best known protocol for circuit satisfiability. This leads to a communication complexity of $\Omega(m^3 \log m)$ bits.

The simplest (non zero-knowledge) proof system for ACP is non-interactive: one just reveals the inputs. So we pay a price for zero-knowledge at least in terms of 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 [28]. In this model, we present protocols for ACP and Boolean Circuit SAT. 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 ACP, the complexity of both our preprocessing and proof phase is $O(m^2)$ bits (the same as for the interactive protocol mentioned above). For the SAT, 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. We note that our *total* communication complexity is the same as that of the best previously known zero-knowledge interactive proofs [8] (which could not be split in a preprocessing and proof phase).

To compare with earlier work on interactive arguments, we need to state the performance of our protocols more accurately: for an error probability of 2^{-k}, and cryptographic security parameter l, the complexity of the preprocessing is $O(ln+k)$ bits. The non-interactive proof stage has size $O(n+l)$. The best earlier work on arguments is by Cramer and Damgård [8] who obtained $O(n)max(l,k)$, and by Kilian [23] 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 [8] and is not directly comparable to [23]. It is superior to [23] 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 [23] 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. Which means that circuits of interest in real applications (say of size 10.000-100.000 gates) would produce proof stages of size 3-26 Kbyte, using a 1024 bit RSA modulus.

Our entire protocol for ACP, resp. the proof stage of our SAT protocol, have the same *worst case* complexity as the simplest non zero-knowledge proof system, where one just reveals the inputs, since indeed this may cost $\Omega(n^2)$, resp. $\Omega(n)$ bits in general. Although our protocols may therefore be optimal in this sense, this does not exclude the possibility of finding much more efficient protocols for particular classes of circuits, e.g. protocols with complexity depending only on the number of inputs. Furthermore, it does not seem impossible to improve the preprocessing for the SAT protocol, e.g. to $O(n)$ bits.

Our final result shows that Shamirs (Shens) [26, 27]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]. So as far as Shen's QBF protocol is concerned, our results do show that zero-knowledge can be for free - but on the other hand, we do not know whether this is an optimal proof system for QBF.

2 Commitment Schemes from Group Homomorphisms

A commitment scheme of the kind we use consists of a function $\texttt{commit} : \{0,1\}^l \times [0..q[\rightarrow \{0,1\}^l$, 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 pair (q, l) 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 the bit length of q is δl.

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 $C \leftarrow \texttt{commit}(r, a)$. To open a commitment, r, a are revealed.

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

For interactive arguments, we use commitment schemes that are *unconditionally hiding*: a commitment to a has distribution independent of a. Then the best we can achieve is that the scheme is *computationally binding*: take any probabilistic polynomial time algorithm which takes an input a public key produced by the generator on input 1^l. Let $\epsilon(l)$ be the probability with which the algorithm outputs a commitment and two valid openings revealing distinct values. Then $\epsilon(l)$ is *negligible*, i.e. for any polynomial p, $\epsilon(l) \le 1/p(l)$ for all large enough l.

2.1 Basic Definitions

Definition 2.1 *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.*

[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. Our result holds for an error probability of 2^n, where n is the input length

$\mathcal{G}$ is said to be one-way *if in addition 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 negligible.*

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 2.2 *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.

Definition 2.3 *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 $0 \le i, j < q$, $i \ne j$ 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.

2.2 Commiment Schemes

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 q-one-way 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 zero-knowledge proof of knowledge that he knows an f-preimage of y. This proof can be easily constructed from the f-preimage protocol in Section 2.3, by using one-bit challenges, and iterating the protocol sequentially.
- **Commitment** to integer $0 \le a < q$: 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 $0 \le a < q$.
- **Hiding Property**: is clear, since if P has accepted the set-up phase, it follows (except possibly with exponentially small probability) that 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 $a > a'$ and $y^a f(r) = y^{a'} f(r')$. 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 2.3. 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 2.3. 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 [20].
- **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 2.3.
- **Binding Property**: Definition 2.3 guarantees that if V accepts the set-up phase, commitments to different values will be in distinct cosets of $Im(f)$.

We will write $[r, a]_y$ for $y^a f(r)$, and sometimes, when no misunderstanding is possible, only $[r, a]$ or $[a]$. It should be clear from the definition of these commitments that both types have a *linear homomorphic property:* given commitments $[r, a]$ and $[s, b]$, P can open $[r, a] \cdot [s, b]$ to reveal $(a + b) \bmod q$. Indeed, 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 q-one wayness, t is easy for P to compute. We have $[r, a] \cdot [s, b] = [rst, (a + b) \bmod q]$. In a similar way, it follows that $[r, a]^c = [r', ca \bmod q]$ and $y^c \cdot [r, a] = [r'', (c + a) \bmod q]$ for a constant c and easily computable (by P) values $r', r'' \in H$.

2.3 Auxiliary Protocols

All protocols in this section are proofs of knowledge and 3-move Arthur-Merlin games, with a random challenge from V as second message. We say that such a protocol has the *special soundness property* if from any pair of conversations $(m, e, z), (m, e', z')$, where $e \neq e'$, one can efficiently compute the information the prover claims to know. Referring to the definition of proofs of knowledge from [3], we have the following which can be found, e.g. in the coming journal version of [13]. It is hardly surprising, but less trivial to prove than one might expect:

Lemma 2.4 *If a protocol has special soundness, it has soundness error $1/c$, where c is the number of possible challenges the verifier chooses from.*

A protocol is *special honest verifier zero-knowledge* (SHVZK), if it has a simulator which on input e produces a correctly distributed conversation (m, e, z).

We first give a 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. The following protocol can be used for any f generated by a q-one-way generator, and is a generalization of Schnorr's discrete log protocol [25]:

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 *challenge* e, so that $0 \leq e < q$ and sends it to P.
3. P sends $z = rv^e$ to V, who checks that $f(z) = mu^e$.

Lemma 2.5 *If P, V follow the protocol, V always accepts. The protocol has the special soundness property and is SHVZK.*

Proof The first claim is trivial. The second follows directly from the definition of q-one-wayness. Finally, on input e, one simulates by choosing at random z and outputting $(f(z)u^{-e}, e, z)$. □

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, 12] to make a protocol in which P proves that C contains 0 or 1, without revealing which is the case. The resulting protocol is referred to as a *bit commitment proof.* It is still SHVZK, and has special soundness. Its communication complexity is $4l + 2\log q$ bits.

The final auxiliary protocol we have is a *multiplication protocol*, an interactive proof showing that the prover can open commitments A, B, C to reveal values a, b, c for which $c = ab \bmod q$. As a side effect, we also obtain a protocol for showing that the prover can open a commitment.

Assume P knows how to write the commitments in the form $A = [r, a]_y$, $B = [u, b]_y$, $C = [s, ab \bmod q]_y$. 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 $C = [t, a]_B$. 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 $ab \bmod q$. With this rewriting of C we see that, loosely speaking, we need a protocol for showing that A contains the same value w.r.t. y as does C w.r.t B. This leads to:

MULTIPLICATION PROTOCOL

Input: f and commitments A, B, C. P knows a, r, t, b, u, such that $A = [r, a]_y$, $C = [t, a]_B$ and $B = [u, b]_y$.

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 [5]:

1. First protocol:
   (a) P chooses $x \in Z_q$ and $s_1, s_2 \in H$ at random and sends $M_1 = [s_1, x]_y$, $M_2 = [s_2, x]_B$ 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 $[w_1, z]_y = M_1 A^e$ and $[w_2, z]_B = M_2 C^e$.
2. Second protocol:
   (a) P chooses $d \in Z_q$ and $s \in H$ at random and sends $M = [s, d]_y$ to V.
   (b) V chooses a random number e, so that $0 \leq e < q$ and sends it to P.

[5] In some cases, the context may imply that P knows how to open B, in which case the second subprotocol can be omitted.

(c) P sets $v = (d + eb) \bmod q$ and chooses j such that $v = d + eb + jq$. He then computes $w = su^e f^{-1}(y^{-jq})$. He sends v, w to V, who verifies that $[w, v]_y = MB^e$

The properties of this protocol are the following:

Lemma 2.6 *If P, V follow the protocol, V always accepts. The protocol has special soundness: from two accepting conversations with challenges e, e', $e \neq e'$, 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 SHVZK.*

Proof The first claim is trivial by inspection. For the second, we let two conversations $(M, M_1, M_2, e, v, w, z, w_1, w_2)$, $(M, M_1, M_2, e', v', w', z', w_1', w_2')$, where $e \neq e'$ be given. If they lead to accept, we immediately obtain 3 equations from each conversation. By dividing them pairwise, we get: $y^{z-z'} f(w_1 {w_1'}^{-1}) = A^{e-e'}$, $B^{z-z'} f(w_2 {w_2'}^{-1}) = C^{e-e'}$ and $y^{v-v'} f(ww'^{-1}) = B^{e-e'}$. Define $\omega = (e-e')^{-1}$ mod q, and i such that $(e-e')\omega = 1 + iq$. Let $\alpha = f^{-1}((B^i)^q)$, which is easy to compute by q-one wayness. Then by raising the last equation to ω, we get

$$B = y^{(v-v')\omega} f((ww'^{-1})^{\omega} \alpha^{-1})$$

which is the desired form. The other two equations can be treated similarly. For honest verifier simulation on input e, choose v, w, z, w_1, w_2 uniformly at random, and compute the rest of the conversation by: $M = y^v f(w) B^{-e}$, $M_1 = y^z f(w_1) C^{-e}$, $M_2 = B^z f(w_2) C^{-e}$. □

The communication complexity of the multiplication protocol is $6l + 3 \log q$ bits.

Both our auxiliary protocols have soundness error $1/q$ by construction. For our main protocols, we will need error 2^{-k}. For this, we will iterate the auxiliary protocols in parallel $\lceil k/\log q \rceil$ times. This works, since SHVZK and special soundness are trivially seen to be preserved under parallel composition.

3 Examples of Group Homomorphism Generators

Any of our generators have 1^l and a prime q as input parameters. Generators with bounded q include as part of their definition a constant δ. Proofs in this section are left to the reader.

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 3.1 *Under the RSA assumption, the RSA generator is a q-one-way generator, with unbounded q.*

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

We now show a generator based on the discrete log problem modulo a prime number. The commitment scheme resulting from this generator was first discovered by Pedersen [24] 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 3.3 *Assume that any probabilistic polynomial time algorithm solves the discrete log problem modulo prime numbers as selected by the Discrete Log Generator with negligible probability. Then the Discrete Log Generator is a q-one-way generator with bounded 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 $f(x) = (g^x \bmod p, h^x \bmod p)$ [6].

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 $(g^r \bmod p, mh^r \bmod p)$ [14]. Recall also the notion of polynomial security, defined by Goldwasser and Micali [18], which says that random encryptions of distinct messages are poynomially indistinguishable.

Lemma 3.4 *If Diffie-Hellman encryption is polynomially secure, then the Diffie-Hellman generator is an unconditionally binding q-homomorphism generator.*

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

4 Protocol Descriptions

This section describes our 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. Well known techniques may then be used to make protocols that are zero-knowledge in general. Common to all our protocols is an *intital step* in which the prover and verifier go through the setup phase for the commitment scheme, as described in Section 2. This can be done once and for all, and the instance of the commitment scheme generated can be reused in several protocol executions. Therefore, we do not mention the intital step explicitly in the descriptions below.

The linear homomorphic property of commitments can 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
V 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 $b_n b_{n-1} ... b_0$. P opens CD^{-1} to reveal 0.

It is easy to see that this game reveals nothing about the value of $b_0, ..b_n$, and that assuming 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.

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

In this section, we are given an arithmetic circuit Ψ over $GF(q)$, where q is an m-bit prime, with u inputs, t multiplication gates, and any number of linear operations. All gates have arbitrary fan-out. We assume for simplicity that there is only one output value computed, from gate G_0, 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.

STEP 1
The prover makes u commitments $I_1, .., I_u$, 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 $x_1, ..., x_u$. All commitments produced are sent to V, and P proves that he knows how to open all of them.

STEP 2
Both P and V compute, based on $I_1, .., I_u, T_1, .., T_t$ and using linearity 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: 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 multiplication protocol to convince V that $ab \bmod q = c$.

PROOF, Step 4
P opens the commitment representing the output value of G_0.

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.

For clarity, we have separated the invocation of subprotocols into steps 1 and 3. However, they can all be executed in parallel, using the same random challenge from V for all of them. By SHVZK for the subprotocols, this can still be simulated against an honest verifier. We get the following, which we state without proof:

Lemma 4.1 *The above protocol is using commitments constructed from a q-one-way generator is perfect honest verifier zero-knowledge, and honest verifier zero-knowledge when using commitments constructed from an unconditionally binding q-homomorphism generator. The communication complexity is $O((u+t)(l+m)\lceil k/m \rceil)$ bits in either case.*

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 $O((u+t)(l+m))$ 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 (will later represent input values), and t random triples of commitments $([d],[e],[f])$ 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 $[a],[b],[c]$ by letting $[a] = y^{\epsilon}[d]$, $[b] = y^{\delta}[e]$ and $[c] = y^{\epsilon+\delta}[f] \cdot [d]^{\delta} \cdot [e]^{\epsilon}$.

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 (where, however, V can handle linear operations on his own).

4.2 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 4.2 *A* NAND-Table *is a matrix with 4 rows and 3 columns containing commitments. A NAND-table is* correct*, if it contains only bit commitments and any of its rows* $([a], [b], [c])$ *satisfies* $a \wedge b = \neg c$*. 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. Then for each row $[a], [b], [c]$, P shows that $1 - c = ab \bmod q$. Assuming that a and b are $0/1$ values, this ensures that so is c, and that $\neg c = a \wedge b$.

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.
Fix a set of input bits that satisfy the circuit. A computation with these input bits selects in a natural way a row in each T_i. For $i = 1..n$, P includes 2 bits in the proof string indicating which row is selected.

Having selected rows in all truth tables, P has defined commitments representing the inputs and output of each gate. He must now demonstrate that "gates connect correctly":

PROOF, Step 2
We make a list of pairs of commitments as follows: Let w be a wire in the circuit. If it connects from T_i to T_j, append to to the list the pair of commitments representing the output from T_i resp. the relevant input to T_j. For each circuit input bit b, let T_k be the first gate receiving b. Append to the list a set of pairs, each of which have the input commitment from T_k as first component and as the second an input commitment from each distinct gate also receiving b.

P must now show that each 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 3
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.

As for ACP, the subprotocols in the preprocessing can be done in parallel. This, and SHVZK for the subprotocols lead to:

Lemma 4.3 *The above protocol using commitments constructed from a q-one-way generator is perfect honest verifier zero-knowledge. If the generator has unbounded q, the communication complexity of the preprocessing is $O(nl + k)$ bits, and $O(n)max(k, l)$ bits otherwise. When using commitments constructed from an unconditionally binding q-homomorphism generator, the protocol is honest verifier zero-knowledge, and the communication complexity of the preprocessing is $O(nl + k)$ bits. The proof stage has size $O(n + l)$ in all cases.*

4.3 Zero-Knowledge Proof for QBF

In [26], Shamir gave the first proof that $IP = PSPACE$, by exhibiting an interactive proof system for the PSPACE complete QBF problem. A little later, Shen [27], 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 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.

We now modify the protocol by having the prover communicate his polynomials by in stead sending commitments to each of the coefficients. This affects the number of bits needed to send a polynomial by at most a constant factor, and furthermore 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 $([a], [b], [c])$, 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. The multiplication protocol allows the prover to convince the verifier about these facts in honest verifier zero-knowledge. Since it is constant round and communicates a constant number of commitments, we get a protocol with the same round and communication complexity, up to a constant factor.

5 Results for the Main Protocols

The results below use the same notation as the corresponding protocol descriptions, and all protocols are designed for an error of 2^{-k}. For formal definitions of proof systems, completeness, soundness and zero-knowledge, please refer to [20]. In the case of arguments, completeness and zero-knowledge are as for proof systems. For computational soundness, we use the definition of [13] (with one change, see below) and show that our protocol, given an instance of the commitment scheme, has soundness error 2^{-k} *relative* to the problem of breaking the commitment scheme. Concretely, this means that if a cheating prover has success probability $\epsilon > 2^{-k}$, then he can break the commitment scheme instance in expected time polynomial in l and linear in $1/(\epsilon - 2^{-k})$. In [13], the circuit to prove is given as input initially. This cannnot be assumed to be true for a protocol with preprocessing. So for this case, we define the success probability of a cheating prover to be the probability with which he can successfully complete the preprocessing, and then compute a non-satisfiable circuit together with a proof that the verifier will accept.

We note that all our communication complexity results are computed without including the complexity of setting up the commitment schemes, since the same commitment scheme instance can be reused in many protocol executions [7].

Theorem 5.1 *If there exists a q-one-way generator with unbounded q then there exists a non-interactive perfect 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, but the communication complexity of the preprocessing becomes $O(n)max(k, l)$ bits.*

Theorem 5.2 *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 $O(n + l)$.*

Theorem 5.3 *If there exists an q-one-way generator, resp. an unconditionally binding q-homomorphism generator then there exists a perfect zero-knowledge*

[7] However, in several cases, including the setup step makes no difference. This is true in general for Theorem 5.1, and for Theorems 5.2, 5.3 when based on the Diffie-Hellman generator.

argument, resp. a computational zero-knowledge proof for ACP. The communication complexity is $O((u+t)(l+m)\lceil k/m \rceil)$ bits in either case.

A sketch of the proofs for these theorems: From Lemmas 4.1, 4.3, we have honest verifier zero-knowledge protocols, which, except for the initial set-up of commitment schemes are 3-move Arthur-Merlin games with k-bit challenges, and have communication complexities as required in the theorems.

To establish soundness, we observe that from correct answers to 2 different challenges, one can compute either a satisfying assignment or two different ways to open some commitment, the latter case being of course impossible with unconditionally binding commitments. This immediately implies soundness for the interactive proof case and, using Lemma 2.4, also for the argument case.

To show zero-knowledge in general, we observe that the interactive arguments we have from the lemmas are already zero-knowledge in general, since the verifier shows knowledge of a trapdoor for the commitments in the initial stage. Adjusting correctly the error probability of this proof, we can ensure that by rewinding the verifier, the simulator can, in expected polynomial time, either extract this trapdoor or exhaustively find a satisfying assignment. Then simulation is trivial in either case. For the interactive proof case, we use the well-known idea that the honest verifier simulator can be used as subrutine in a real simulation provided that the verifier commits to his challenge in advance. For a solution of the subtle technical problems with this, see [17]. If we use our unconditionally hiding commitments for this part, both soundness and asymptotic communication complexity will be unaffected.

Theorem 5.4 *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 4.3 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 4.3, but can be made zero-knowledge with no essential loss of efficiency in the same way as above, using the method from [17]. 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]. 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 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. □

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 Polylogarithmic 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.
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. R. Cramer and I. Damgård: *Linear Zero-Knowledge*, Proc. of STOC 97.
9. R. Cramer, I. Damgård and U. Maurer: *Span Programs and General Secure Multiparty Computations*, BRICS Report series RS-97-27, available from http://www.brics.dk.
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. De Santis, Di Crecenzo, Persiano and Yung, Proceedings of FOCS 1994.
13. I. Damgaård and B. Pfitzmann: *Sequential Iteration of Interactive Arguments*, Proc. of ICALP 98, Springer Verlag LNCS series.
14. 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.
15. L.Fortnow: *The complexity of Perfect Zero-Knowledge*, Adv. in Computing Research, vol.5, 1989, 327–344.
16. E. Fujisaki and T. Okamoto: *Statistical Zero-Knowledge Protocols to prove Modular Polynomial Relations*, Proceedings of Crypto 97, Springer Verlag LNCS series.
17. O. Goldreich and A. Kahan: *How to Construct Constant-Round Zero-Knowledge Proof Systems for NP*, Journal of Cryptology, (1996) 9: 167–189.
18. S. Goldwasser and S. Micali: *Probabilistic Encryption*, JCSS, vol.28, 1984.
19. 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.
20. S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, Vol. 18, pp. 186-208, 1989.
21. R.Gennaro, T.Rabin and M.Rabin: *Simplified VSS and Fast-Track Multiparty Computations*, Manuscript, 1997.
22. J. Kilian: *A note on Efficient Proofs and Arguments*, Proceedings of STOC '92.
23. J. Kilian: *Efficient Interactive Arguments*, Proceedings of Crypto '95, Springer Verlag LNCS, vol. 963, pp. 311–324.
24. T. Pedersen: *Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing*, proc. of Crypto 91, Springer Verlag LNCS, vol. 576, pp. 129–140.
25. C. P. Schnorr: *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, 4 (3): 161–174, 1991.
26. A.Shamir: *IP=PSPACE*, Journal of the ACM, vol.39 (1992), 869-877.
27. A. Shen: *IP=PSPACE, Simplified Proof*, Journal of the ACM, vol.39 (1992), pp.878-880.
28. 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.