

On the Amortized Complexity of Zero Knowledge Protocols for Multiplicative Relations

Ronald Cramer, Ivan Damgård and Valerio Pastro

CWI Amsterdam and Dept. of Computer Science, Aarhus University

Abstract. We present a protocol that allows to prove in zero-knowledge that committed values $x_i, y_i, z_i, i = 1, \dots, l$ satisfy $x_i y_i = z_i$, where the values are taken from a finite field. For error probability 2^{-u} the size of the proof is linear in u and only logarithmic in l . Therefore, for any fixed error probability, the amortized complexity vanishes as we increase l . In particular, when the committed values are from a field of small constant size, we improve complexity of previous solutions by a factor of l . Assuming preprocessing, we can make the commitments (and hence the protocol itself) be information theoretically secure. Using this type of commitments we obtain, in the preprocessing model, a perfect zero-knowledge interactive proof for circuit satisfiability of circuit C where the proof has size $O(|C|)$. We then generalize our basic scheme to a protocol that verifies l instances of an algebraic circuit D over K with v inputs, in the following sense: given committed values $x_{i,j}$ and z_i , with $i = 1, \dots, l$ and $j = 1, \dots, v$, the prover shows that $D(x_{i,1}, \dots, x_{i,v}) = z_i$ for $i = 1, \dots, l$. The interesting property is that the amortized complexity of verifying one circuit only depends on the multiplicative depth of the circuit and not the size. So for circuits with small multiplicative depth, the amortized cost can be asymptotically smaller than the number of multiplications in D . Finally we look at commitments to integers, and we show how to implement information theoretically secure homomorphic commitments to integer values, based on preprocessing. After preprocessing, they require only a constant number of multiplications per commitment. We also show a variant of our basic protocol, which can verify l integer multiplications with low amortized complexity. This protocol also works for standard computationally secure commitments and in this case we improve on security: whereas previous solutions with similar efficiency require the strong RSA assumption, we only need the assumption required by the commitment scheme itself, namely factoring.

1 Introduction

The notions of commitment schemes and zero-knowledge proofs are among the most fundamental in the theory and practice of cryptographic protocols. Intuitively, a commitment scheme provides a way for a prover to put a value x in a locked box and commit to x by giving this box $[x]$ to a verifier. Later the prover can choose to open the box by giving away the key to the box.

In a zero-knowledge protocol, a prover wants to convince a verifier that some statement is true, such that the verifier learns nothing except the validity of the assertion. Typically, the prover claims that an input string u is in a language L , and after the interaction, the verifier accepts or rejects. We assume the reader is familiar with the basic theory of zero-knowledge protocols and just recall the most important notions informally: the protocol is an interactive zero-knowledge proof system for L if it is *complete*, i.e., if $u \in L$, then the verifier accepts – and *sound*, i.e., if $u \notin L$ then no matter what the prover does, the verifier accepts with at most probability ϵ , where ϵ is called the soundness error of the protocol. Finally, zero-knowledge means that given only that $u \in L$, conversations between the honest prover and an arbitrary poly-time verifier can be efficiently simulated and are indistinguishable from real conversations.

In this paper we concentrate on commitments to elements in a finite field K , or to integers and we assume that commitments are also homomorphic, i.e., both commitments and randomness are chosen from (finite) groups, and $[x] \cdot [y] = [x+y]$ (we will describe this property more in detail in section 2.1 and 7). For $K = \mathbb{F}_q$ for a prime q , such commitments can, for instance, be constructed from any q -invertible group homomorphism [CD98] that exists, if factoring or discrete log are hard problems. It is also easy based on known techniques – but perhaps less well known – that homomorphic commitments with unconditional hiding *and* binding can

be built if we assume preprocessing, e.g., the committer gets random field elements and information theoretic MACs and the receiver gets corresponding keys. We give more details on this later (see Section 2.1). Finally, Homomorphic commitments to integers based on factoring were proposed in [FO97,DF02].

In typical applications of these commitment schemes, the prover needs to convince the verifier that the values he commits to satisfy a certain algebraic relation. A general way to state this is that the prover commits to x_1, \dots, x_v , and the verifier wants to know that $D(x_1, \dots, x_v) = 0$ for an algebraic circuit D defined over K or over the integers. If D uses only linear operations, the verifier can himself compute a commitment to $D(x_1, \dots, x_v)$ (using the homomorphic properties of the commitment scheme) and the prover opens this to reveal 0. However, if D uses multiplication, we need a zero-knowledge protocol where the prover convinces the verifier that three committed values x, y, z satisfy $xy = z$.

In [CDD⁺99], such a multiplication protocol was proposed for homomorphic commitments over any finite field K . The soundness error for that protocol is $1/|K|$, which is too large if K is a field with small size (constant or logarithmic in the security parameter). The only known way to have a smaller error is to repeat the protocol. This solution leads to a protocol with communication complexity $\Theta(\kappa l)$ for soundness error 2^{-l} and where commitments have size κ bits.

Likewise, a multiplication protocol for integer commitments was proposed in [FO97,DF02]. This protocol has essentially optimal communication complexity $\Theta(\kappa + l + k)$, where k is size in bits of the prover's secret integers, but it requires an extra assumption, namely the strong RSA assumption. If we only want to assume what the commitment scheme requires (factoring), the best known complexity is $\Theta((\kappa + k)l)$.

An approach to improving this state of affairs was proposed in [CD09], where it was suggested to take advantage of the fact that many applications require the prover to make many ZK proofs of similar statements. The idea is to make the amortized complexity per proof be small by combining all the proofs into one protocol. In our case, this would mean that the prover commits to x_i, y_i, z_i for $i = 1, \dots, l$ and wants to convince the verifier that $x_i y_i = z_i$ for all i . The technique from [CD09] yields a protocol with amortized complexity $\Theta(\kappa + l)$ but, unfortunately, requires that all x_i 's are equal (or all y_i 's are equal), and in most applications, this condition is not satisfied.

1.1 Our Contribution

In this paper, we construct a new zero-knowledge protocol that works for arbitrary x_i, y_i, z_i , and uses black-box access to any homomorphic commitment scheme. If we instantiate the commitments by a standard unconditionally binding and computationally hiding scheme, the amortized complexity is $O(\frac{u}{l} \cdot \kappa)$ bits for error probability 2^{-u} . In particular, for $l = u$, we get $O(\kappa)$. Therefore, when the committed values are from a field of small constant size, we improve the complexity of previous solutions by a factor of l . We also propose (based on standard techniques) a way to implement unconditionally secure homomorphic commitments assuming preprocessing. Using this implementation, the amortized complexity is $O(\frac{u}{l} \cdot u)$. In particular, for for both types of commitments and any fixed error probability, the amortized overhead vanishes as we increase l .

We generalize our approach to obtain a protocol that verifies l instances of an algebraic circuit D over K with v inputs, in the following sense: given committed values $x_{i,j}$ and z_i , with $i = 1, \dots, l$ and $j = 1, \dots, v$, the prover shows that $D(x_{i,1}, \dots, x_{i,v}) = z_i$ for $i = 1, \dots, l$ (the protocol easily generalizes to circuits with more than one output). The amortized cost to verify one circuit with multiplicative depth δ is $O(2^\delta \kappa + v\kappa + \delta \log l)$ bits for an error probability of 2^{-l} and so does not depend on the circuit size. For circuits with small multiplicative depth (sometimes known as the classes K -SAC⁰ or K -SAC¹), this approach is better than using our first protocol, in fact the amortized communication cost can be asymptotically smaller than the number of multiplications in D .

Another interesting feature of this protocol is that prover and verifier can execute it given only black-box access to an algorithm computing the function implemented by D . This is in contrast to standard protocols where the parties work their way through the circuit and must therefore agree on the layout. Our protocol would, for instance, allow the verifier to outsource computation of the function to a third party. As long as the verifier chooses the random challenge in the protocol, this would be secure if the prover is malicious and the third party is semi-honest.

Our final result is a zero-knowledge protocol using black-box access to homomorphic commitments to k -bit integers. For checking l integer multiplications and error probability 2^{-l} , the amortized complexity is $O(\kappa + k + l \log(l))$. When instantiating the commitments using a standard computationally secure scheme, this improves security of previous solutions that needed the strong RSA assumption, while we need no assumption, other than what the underlying commitment scheme requires (typically factoring). We also show a new technique for implementing unconditionally secure homomorphic commitments to integers based on preprocessing. This makes the protocol be much more efficient, as only a constant number of multiplications per commitment is required.

When using information theoretically secure commitments based on preprocessing, our protocols are perfect zero-knowledge against general verifiers. When using standard computationally secure commitments, they are only honest verifier zero-knowledge, but can be made zero-knowledge in general using standard techniques.

Our technique is somewhat related to the “MPC-in-the-head” technique from [IKOS09], but with an important difference: both strategies make use of “virtual players”, that is, the prover in his head imagines n players that receive shares of his secret values and he must later reveal information to the verifier relating to these shares. The protocol from [IKOS09] has complexity linear in n , because the prover must commit to the view of each virtual player. We use a different approach, exploiting the homomorphic property of the commitment scheme to get a protocol with complexity logarithmic in n . This is the reason our amortized overhead vanishes instead of being constant, as one would get using MPC-in-the-head. On the other hand, we show that a combination of “multiparty computation in the head” and our protocol for verifying algebraic circuits can actually improve the communication complexity for some parameter values. In concurrent and independent work, Ben-Sasson et al. [BSFO11] show a multiparty protocol for honest majority that checks several multiplicative relations on secret-shared values with low amortized complexity. The technique is somewhat related in that it is based on secret sharing, but the checking works in a different way since in that setting there is no single prover who knows all values.

1.2 Applications

One obvious application of our protocol is to give ZK proofs for satisfiability of a Boolean circuit C : the prover commits to the bits on each wire in the circuit, opens the output as a 1 and shows that, for each AND-gate, the corresponding multiplicative relation holds for the committed bits. To explain how this compares to previous work, we define the (computational or communication) overhead of a protocol to be its (computational or communication) complexity divided by $|C|$. One can think of this as the overhead factor one has to pay to get security, compared to an insecure implementation. Now, in the ideal commitment model (i.e., assuming access to a ideal commitment functionality) [IKOS09] obtained constant communication overhead and polynomial computation overhead, as a function of the security parameter u , for error probability 2^{-u} . Later, [DIK10] showed how to make both overheads poly-logarithmic. For both protocols, the ideal commitments can be implemented by doing preprocessing, and the resulting “on-line” protocol will still have the same complexity.

As mentioned, our protocol can be thought of as working in the ideal *homomorphic* commitment model where the commitment functionality can do linear operations on committed bits (but where we of course charge for the cost of these operations). In this model our protocol achieves constant computational and communication overhead.

We may then instantiate the commitments using the information theoretically secure homomorphic scheme. This incurs an extra cost for local computing, so as a result we obtain a ZK-protocol with constant communication overhead and polynomial computation overhead (essentially $O(u \log u)$). Asymptotically, the overheads match those of [IKOS09], but the involved constants are smaller in our case because we do not need the “detour” via a multiparty protocol. Finally, we pay no communication for linear operations, while this seems hard to achieve in the protocol from [IKOS09].

Another application area where our result can improve state of the art is the following: as shown in [CDN01], general multiparty computation can be based on additively homomorphic encryption schemes. Many such schemes are known, and in several cases, the plaintext space is a small field. One example is the Goldwasser-Micali (GM)-scheme [GM84], where the plaintext space is \mathbb{F}_2 . Supplying inputs to such a protocol

amounts to sending them in encrypted form to all players and proving knowledge of the corresponding plaintexts. However, in many applications one would want to check that inputs satisfy certain conditions, e.g., an auction may require that bids are numbers in a certain interval. Since ciphertexts in such an additively homomorphic scheme can be thought of as homomorphic commitments over the field, our protocol can be used by a player to prove that his input satisfy a given condition much more efficiently than by previous techniques.

A final type of application is in the area of anonymous credentials and group signatures. Such constructions are often based on zero-knowledge proofs that are made non-interactive using the Fiat-Shamir heuristic. If the proof requires showing that a committed number is in a given interval, the standard solution is to “transfer” the values to an integer commitment scheme and use the proof technique of Boudot [Bou00]. This in turn requires multiplication proofs, so if sufficiently many proofs are to be given in parallel, one can use our technique for integer commitments. Assuming preprocessing and our information theoretically secure commitments this can be very efficient, requiring only a constant number of multiplications per commitment.

2 Preliminaries

2.1 Information Theoretic Commitments

In this section we assume a setup that allows commitments to be unconditionally secure. We use $[v]$ as shorthand for a commitment to v in the following. Operations on commitments are supposed to be multiplicative, while values that are committed lie in an additive group. Therefore a commitment scheme is homomorphic if $[v] \cdot [v'] = [v + v']$ for all v, v' in the proper domain (either a finite field K or the integers). Also, if $\mathbf{v} = (v_1, \dots, v_m)$ is a vector with entries in K (or in the integers), $[\mathbf{v}]$ denotes a vector of commitments, one to each coordinate in \mathbf{v} . If $\mathbf{u} = (u_1, \dots, u_m)$ is a vector of the same length as \mathbf{v} , then $[\mathbf{v}]^{\mathbf{u}}$ means $[\mathbf{v}]^{\mathbf{u}} = \prod_i [v_i]^{u_i}$, which is a commitment containing the inner product of \mathbf{u} and \mathbf{v} . Moreover $[\mathbf{u}] * [\mathbf{v}]$ refers to the component-wise product.

Field Scenario Let K be a finite field and L be an extension of K . Although the set-up is general, we will think of K as a small constant size field in the following. Let $a \in L$ be a private value held by the verifier. We suppose that the prover has a list A of uniform values $u_1, \dots, u_i, \dots \in K$ and for each u_i he also has a value $m_{u_i} = a \cdot u_i + b_{u_i}$, where b_{u_i} is uniform in L and privately held by the verifier. One can think of m_{u_i} as an information theoretic message authentication code on u_i , and of (a, b_{u_i}) as the key to open such a MAC on u_i . It is possible to achieve this situation assuming a functionality for the preprocessing phase of a multiparty computation protocol, such as in the ones in [BDOZ11, DPSZ12].

With this setup, commitments can be done as follows: In order for the prover to commit to $v \in K$, the prover sends $u - v$ to the verifier and sets $m_v = m_u = a \cdot u + b_u$, where u is the first unused value in the list A ; the verifier then updates the corresponding key b_u into $b_v = b_u + a \cdot (u - v)$. A commitment to v can therefore be expressed as the following data (where P denotes the prover, and V denotes the verifier);

$$[v] = \begin{cases} P : v, u, m_v = a \cdot u + b_u \\ V : u - v, a, b_v = b_u + a \cdot (u - v) \end{cases}$$

In order to open commitment $[v]$, the prover sends v, m_v to the verifier, who checks if $a \cdot v + b_v$ equals m_v .

Commitments of this form are unconditionally binding: A prover committing to v can send an opening \tilde{v}, \tilde{m} with $\tilde{v} \neq v$ if and only if $a \cdot \tilde{v} + b_v = \tilde{m}$. This is equivalent for the prover to be able to sample two distinct points $(P_x, P_y) = (v, m_v), (Q_x, Q_y) = (\tilde{v}, \tilde{m})$ from the line $Y = a \cdot u + b_v$; which is equivalent for the prover to know the key (a, b_v) privately held by the verifier. This shows that the probability of a prover succeeding in opening to a different value is bounded by the probability of guessing a random element in a line over L ; such a probability equals $1/|L|$. Since we want to have a negligible probability of breaking the binding property of the commitment scheme, we require $|L| = 2^{\Theta(\kappa)}$, where κ is the security parameter. In particular, this means that in (the unfortunate) case where K is a field with constant size, then L is an extension of K of degree linear in κ .

These commitments are also unconditionally hiding: A verifier receiving a commitment $u - v$ only knows a and $b_{u_1}, \dots, b_{u_i}, \dots$, which are all independent from v .

Moreover, the above commitments are homomorphic (meaning: $[v] \cdot [v'] = [v + v']$), where $[v] \cdot [v']$ is defined as follows:

$$[v] \cdot [v'] := \begin{cases} P : v + v', u + u', m_{v+v'} = m_u + m_{u'} \\ V : (u - v) + (u' - v'), a, b_{v+v'} = b_u + b_{u'} + a \cdot (u - v + u' - v') \end{cases}$$

Integers Scenario We here give a construction of unconditionally secure commitments on k -bits integers. Contrary to the previous construction, this one is new, to the best of our knowledge. Let a be a prime in the interval $[-2^\kappa, \dots, 2^\kappa]$ privately held by the verifier. We assume the prover has a list A of integer values u_1, \dots, u_i, \dots uniform in $[-2^{k+\kappa}, \dots, 2^{k+\kappa}]$ and for each u_i he also has an integer $m_{u_i} = a \cdot u_i + b_{u_i}$, where b_{u_i} is a uniform integer in $[-2^{k+3\kappa}, \dots, 2^{k+3\kappa}]$ and privately held by the verifier.

With this setup, commitments can be done as follows: In order for the prover to commit to the integer $v \in [-2^k, \dots, 2^k]$, the prover sends $u - v$ to the verifier and sets $m_v = m_u = a \cdot u + b_u$, where u is the first unused value in the list A ; the verifier then updates the corresponding key b_u into $b_v = b_u + a \cdot (u - v)$. A commitment to v can therefore expressed as the following data (where P denotes the prover, and V denotes the verifier);

$$[v] = \begin{cases} P : v, u, m_v = a \cdot u + b_u \\ V : u - v, a, b_v = b_u + a \cdot (u - v) \end{cases}$$

In order to open commitment $[v]$, the prover sends v, m_v to the verifier, who checks if $a \cdot v + b_v$ equals m_v .

Commitments of this form are homomorphic, unconditionally hiding (same arguments as above) and unconditionally binding: A prover committing to v can send an opening \tilde{v}, \tilde{m} with $\tilde{v} \neq v$ if and only if $a \cdot \tilde{v} + b_v = \tilde{m}$. Subtracting the latter equation to the relation $m_v = a \cdot v + b_v$, we obtain $\tilde{m} - m_v = a \cdot (\tilde{v} - v)$, so the prover must know a multiple of a of length $k + \kappa$ bits. Any $(k + \kappa)$ -bits integer can be thought of its factorization, and the prover can break the binding property if he knows a $(k + \kappa)$ -bits integer where a appears in its factorization. Since a $(k + \kappa)$ -bits integer contains at most $(k + \kappa)/\kappa$ prime factors of length κ and the number of κ -bits primes is $\Theta(2^\kappa/\kappa)$, then the error probability of the scheme is equal to $\Theta(((k + \kappa)/\kappa) \cdot (\kappa/2^\kappa)) = \Theta((k + \kappa)/2^\kappa)$. If $k = O(\kappa)$, the error probability is $O(\kappa/2^\kappa)$.

2.2 Linear Secret Sharing Schemes

The model of linear secret sharing schemes we consider here is essentially equivalent to both the monotone span program formalism [KW93,CDM00] and the linear code based formalism [CCG⁺07]. However, we generalize to schemes where several values from the underlying field can be shared simultaneously. The model is designed to allow us to describe our protocol to follow as easily as possible.

Let K be a finite field and let m be a positive integer. Consider the m -dimensional K -vector space K^m . Consider the index set $I = \{1, 2, \dots, m\}$, and write $\mathbf{x} = (x_i)_{i \in I}$ for the coordinates of $\mathbf{x} \in K^m$. In the following, linear functions between finite spaces are considered. It is useful to recall that because such functions are (additive) group homomorphisms, they are always regular; that is, each element in the image has the same number of pre-images, namely the cardinality of the kernel.

For a non-empty set $A \subseteq I$, the **restriction** to A is the K -linear function

$$\begin{aligned} \pi_A : K^m &\longrightarrow K^{|A|} \\ \mathbf{x} &\longmapsto (x_i)_{i \in A}. \end{aligned}$$

Let $C \subseteq K^m$ be a K -linear subspace which we keep fixed throughout this section. Let $A, S \in I$ be non-empty sets. We say that S **offers uniformity** if $\pi_S(C) = K^{|S|}$. Note that by regularity of π_S , if \mathbf{c} is uniform in C , then $\pi_S(\mathbf{c})$ is uniform in $K^{|S|}$.

Jumping ahead, we will use the subspace C for secret sharing by choosing a random vector $\mathbf{c} \in C$ such that $\pi_S(\mathbf{c}) = \mathbf{s}$ where S is a set offering uniformity and \mathbf{s} is the vector of secret values to be shared. The shares are then the coordinates of \mathbf{c} that are not in S .

We say that A **determines** S if there is a function $f : K^{|A|} \rightarrow K^{|S|}$ such that, for all $\mathbf{c} \in C$, $(f \circ \pi_A)(\mathbf{c}) = \pi_S(\mathbf{c})$. Note that such f is K -linear if it exists. Note that if \mathbf{c} is uniformly chosen from C and if A determines S , then $\pi_A(\mathbf{c})$ determines $\pi_S(\mathbf{c})$ with probability 1.

We say that A and S are **mutually independent** if the K -linear function

$$\begin{aligned} \phi_{A,S} : C &\longrightarrow \pi_A(C) \times \pi_S(C) \\ \mathbf{c} &\longmapsto (\pi_A(\mathbf{c}), \pi_S(\mathbf{c})) \end{aligned}$$

is surjective. Note that $\pi_S(C) = \{\mathbf{0}\}$ is the only condition under which it occurs that both A and S are independent and A determines S . In particular, if \mathbf{c} is uniformly chosen from C , then $\pi_S(C) \neq \{\mathbf{0}\}$ and if A and S are independent, then $\pi_A(\mathbf{c})$ and $\pi_S(\mathbf{c})$ are distributed independently.

Suppose S offers uniformity. Let e be a positive integer and let

$$g : K^{|S|+e} \longrightarrow C$$

be a surjective K -linear function. Define $\pi_g : K^{|S|+e} \rightarrow K^{|S|}$ as the projection to the first $|S|$ coordinates. We say that g is an S -**generator** for C if $\pi_g = \pi_S \circ g$, that is, if the first $|S|$ coordinates of $\rho \in K^{|S|+e}$ are the same as the coordinates of $g(\rho)$ designated by S . Such an S -generator always exists, by elementary linear algebra, with $|B| + \rho = \dim_K(C)$.

For any S -generator g we have that if $\mathbf{s} \in K^{|S|}$ is fixed and if $\rho_{\mathbf{s}}$ is uniformly chosen in $K^{|S|+e}$ subject to $\pi_g(\rho_{\mathbf{s}}) = \mathbf{s}$, then $g(\rho_{\mathbf{s}})$ has the uniform distribution on the subset of C consisting of those $\mathbf{c} \in C$ with $\pi_S(\mathbf{c}) = \mathbf{s}$.

We are now ready to define linear secret sharing schemes in our model: Let $S \subset I$ be non-empty and proper. Write $S^* = I \setminus S$. The tuple (C, S) is a **linear secret sharing scheme** if S offers uniformity and if S^* determines S .

If that is the case, S^* is called the **player set**, $\pi_S(C)$ is the **secret-space**, and $\pi_{S^*}(C)$ is the **share-space**. If $j \in S^*$, then $\pi_j(C)$ is called the **share-space** for the j -th player. If $l = |S|$, the scheme is said to be l -**multi-secret**. For $A \subseteq S^*$, we say that the scheme has A -**privacy** (or A is an unqualified set) if $A = \emptyset$ or if A and S are independent. There is A -**reconstruction** (or A is qualified) if A is non-empty and if A determines S . The scheme offers t -**privacy** if, for all A in the player set with $|A| = t$, there is A -privacy. The scheme offers r -**reconstruction** if, for all A in the player set with $|A| = r$, there is A -reconstruction. Note that $0 \leq t < r \leq |S^*|$ if there is t -privacy and r -reconstruction. A **generator** for (C, S) is an S -generator for C .

Let (C, S) be a secret sharing scheme, and let g be a generator. If $\mathbf{s} \in K^{|S|}$ is the secret, shares for the players in S^* are computed as follows. Select a vector $\rho_{\mathbf{s}}$ according to the uniform probability distribution on $K^{|S|+e}$, subject to $\pi_g(\rho_{\mathbf{s}}) = \mathbf{s}$ and compute $\mathbf{c} = g(\rho_{\mathbf{s}})$. The “full vector of shares” is the vector $\pi_{S^*}(\mathbf{c})$.

In the following, where we write $\rho_{\mathbf{s}}$, it will usually be understood that it holds that $\pi_g(\rho_{\mathbf{s}}) = \mathbf{s}$, and we say that such a vector is consistent with the secret \mathbf{s} .

Multiplication Properties For any $\mathbf{x}, \mathbf{y} \in K^m$, the **Schur-product** (or component-wise product) between them is the element $(\mathbf{x} * \mathbf{y}) \in K^m$ defined as $(\mathbf{x} * \mathbf{y}) = (x_j \cdot y_j)_{j \in I}$. If $C \subset K^m$ is a K -linear subspace, then its **Schur-product transform** is the subspace $\widehat{C} \subset K^m$ defined as the K -linear subspace generated by all elements of the form $\mathbf{c} * \mathbf{c}'$, where $\mathbf{c}, \mathbf{c}' \in C$.

Note that if (C, S) is a linear secret sharing scheme, then S offers uniformity in \widehat{C} as well. But in general it does not hold that S^* determines S in \widehat{C} . However, suppose that it does (so (\widehat{C}, S) is a linear secret sharing scheme). Then (C, S) is said to offer \widehat{r} -**product reconstruction** if (\widehat{C}, S) offers \widehat{r} -reconstruction.

Sweeping vectors Let (C, S) be a linear secret sharing scheme, let g be a generator for it and let A be an unqualified set. Since A and S are mutually independent so that $\phi_{A,S}$ is surjective, it follows that for any index $j \in S$, there exists $\mathbf{c}_{A,j} \in C$ such that $\phi_{A,S}(\mathbf{c}_{A,j}) = (\mathbf{0}, \mathbf{e}_j)$ where \mathbf{e}_j is the vector with a 1 in

position j and zeros elsewhere. Note that since the generator g is surjective on C we can choose $\mathbf{w}_{A,j}$ such that $g(\mathbf{w}_{A,j}) = \mathbf{c}_{A,j}$, and $\pi_g(\mathbf{w}_{A,j}) = \mathbf{e}_j$. The vector $\mathbf{w}_{A,j}$ is called a **j th sweeping vector**.

To see the purpose of these vectors, suppose we have shared a vector of $|S|$ zeros, so we have $\mathbf{c}_0 = g(\rho_0)$. It is now easy to see that the vector

$$\rho_0 + \sum_{j=1}^{|S|} x_j \mathbf{w}_{A,j}$$

is consistent with the secret $(x_1, \dots, x_{|S|})$. Moreover, if we apply g to this vector, the player set A gets the same shares as when 0's were shared.

3 Our Protocol

We are now ready to solve the problem mention in the introduction, namely the prover holds values $\mathbf{x} = (x_1, \dots, x_l)$, $\mathbf{y} = (y_1, \dots, y_l)$, $\mathbf{z} = (z_1, \dots, z_l)$, has sent commitments $[\mathbf{x}], [\mathbf{y}], [\mathbf{z}]$ to the verifier and now wants to convince the verifier that $x_i y_i = z_i$ for $i = 1, \dots, l$, i.e., that $\mathbf{x} * \mathbf{y} = \mathbf{z}$.

We suppose that both the prover and the verifier agreed on using an l -multisecret linear secret sharing scheme (C, S) , for d players, offering \hat{r} -product reconstruction, and with privacy threshold t . We fix a generator $g : K^{l+e} \rightarrow C$. Moreover, we suppose that $\hat{g} : K^{l+\hat{e}} \rightarrow \hat{C}$ is a generator for (\hat{C}, S) and that a public basis for K^{l+e} (respectively for $K^{l+\hat{e}}$) has been chosen such that the linear mapping g (resp. \hat{g}) can be computed as the action of a matrix M (resp. \hat{M}).

The idea of the protocol is as follows: the prover secret shares \mathbf{x} and \mathbf{y} using (C, S) and \mathbf{z} using (\hat{C}, S) , in such a way that the resulting vectors of shares $\mathbf{c}_x, \mathbf{c}_y, \hat{\mathbf{c}}_z$ satisfy $\mathbf{c}_x * \mathbf{c}_y = \hat{\mathbf{c}}_z$, which is possible since (C, S) offers product reconstruction. The prover commits to the randomness used in all sharings, which, by the homomorphic property, allows the verifier to compute commitments to any desired share. The verifier now chooses t coordinate positions randomly and asks the prover to open the commitments to the shares in those positions. The verifier can then check that the shares in \mathbf{x}, \mathbf{y} multiply to the shares in \mathbf{z} . This is secure for the prover since any t shares reveal no information, but on the other hand, if the prover's claim is false, thus $\mathbf{x} * \mathbf{y} \neq \mathbf{z}$, then $\mathbf{c}_x * \mathbf{c}_y$ and $\hat{\mathbf{c}}_z$ can be equal in at most \hat{r} positions, so the verifier has a good chance of finding a position that reveals the cheat. More formally, the protocol goes as follows:

Protocol Verify Multiplication

1. The prover chooses two vectors $\mathbf{r}_x, \mathbf{r}_y \in K^e$, and sets $\rho_x = (\mathbf{x}, \mathbf{r}_x)$, $\rho_y = (\mathbf{y}, \mathbf{r}_y)$. Define $\mathbf{c}_x = M\rho_x$, $\mathbf{c}_y = M\rho_y$. Now, the prover computes $\hat{\rho}_z \in K^{l+\hat{e}}$ such that $\hat{\rho}_z$ is consistent with secret \mathbf{z} and such that $\hat{M}\hat{\rho}_z = \mathbf{c}_x * \mathbf{c}_y$.
Note that this is possible by solving a system of linear equations, exactly because $\mathbf{x} * \mathbf{y} = \mathbf{z}$. We then write $\hat{\rho}_z = (\mathbf{z}, \hat{\mathbf{r}}_z)$ for some $\hat{\mathbf{r}}_z \in K^{\hat{e}}$. Set $\hat{\mathbf{c}}_z = \hat{M}\hat{\rho}_z$.
2. The prover sends vectors of commitments $[\mathbf{r}_x], [\mathbf{r}_y], [\hat{\mathbf{r}}_z]$ to the verifier. Together with the commitments to \mathbf{x}, \mathbf{y} and \mathbf{z} , the verifier now holds vectors of commitments $[\rho_x], [\rho_y], [\hat{\rho}_z]$.
3. The verifier chooses t uniform indices $O \subset S^*$ and sends them to the prover.
4. Let \mathbf{m}_i be the i 'th row of M and $\hat{\mathbf{m}}_i$ the i 'th row of \hat{M} . For each $i \in O$, using the homomorphic property of the commitments, both prover and verifier compute commitments

$$[(\mathbf{c}_x)_i] = [\rho_x]^{\mathbf{m}_i}, \quad [(\mathbf{c}_y)_i] = [\rho_y]^{\mathbf{m}_i}, \quad [(\hat{\mathbf{c}}_z)_i] = [\hat{\rho}_z]^{\hat{\mathbf{m}}_i}.$$

The prover opens these commitments to the verifier.

5. The verifier accepts if and only if the opened values satisfy $(\mathbf{c}_x)_i \cdot (\mathbf{c}_y)_i = (\hat{\mathbf{c}}_z)_i$ for all $i \in O$.

Theorem 1. *Assume the commitment scheme used is the one described in section 2.1. Then protocol Verify Multiplication is perfect zero-knowledge, and if for some i , $x_i y_i \neq z_i$, the verifier accepts with probability at most $((\hat{r} - 1)/d)^t + 1/|L|$.*

In the appendix we give a proof for the above theorem ¹. Theorem 4 given later covers the case where we use standard computationally based commitments.

Representing a Sequence of Points We here compare two approaches to represent a sequence O of t elements drawn uniformly and independently from a set $S^* = \{1, \dots, d\}$. This comparison helps us to find the best communication complexity achievable by our protocols, since in all of them there is one step in which such O has to be sent between players.

One method is to send the sequence of points in O : such a procedure requires $\log d$ bits per point; so the total amount of bits is $t \cdot \log d$. An alternative method is to send a bit-vector of dimension d , where its i th coordinate is equal to 1 if and only if $i \in O$; the total amount of bits is then d .

In the proceeding version of our paper, we always assumed that the communication between players was performed using the first method, while in the full version we take advantage of the most convenient method: when the choice of the parameters is such that $d < t \cdot \log d$ holds, we use the latter (in the protocol for verification of a single multiplication, both for finite fields and for the integers), while we use former when the inequality is not satisfied (in the protocol for verification of a circuit).

Demands to the Secret Sharing Schemes. Above, we have described the protocol for a fixed secret sharing scheme, but what we really want is to look at is the asymptotic behavior as a function of l , the number of secrets we handle in one execution, and u , where we want error probability 2^{-u} . For this, we need a family of secret sharing schemes, parametrized by l, u , which will make t, d, e, \hat{r} and \hat{e} be functions of l, u .

Say that committing requires sending κ_c bits while opening requires κ_o bits. For standard computationally secure commitments, it is usually the case that κ_c is $\Theta(\kappa_o)$, but this is not the case for the information theoretically secure commitments, where κ_c can be much smaller than κ_o . Using this notation, the communication complexity of the protocol is $O(\kappa_c(e + \hat{e}) + \kappa_o t + d)$ bits.

Now, suppose we can build a family of secret sharing schemes, where e, \hat{e} are $O(u)$ and \hat{r} is $O(l + u)$, t is $\Theta(u)$ and $(\hat{r} - 1)/d$ is $O(1)$. This allows d to be $O(l + u)$ and so we can achieve the complexities we promised earlier: For standard computationally hiding commitments, we get we get soundness error $2^{-\alpha u}$ for some constant $\alpha > 0$ for one instance of the protocol. For the information theoretically secure commitments, we get the same if we set $|L| = 2^{\Theta(u)}$. In any case, if necessary, we can achieve 2^{-u} by repeating in parallel a constant number of times. Inserting in the above expression, and dividing by l , we get the complexity per multiplicative relation: $O(\frac{u}{l}(\kappa_c + \kappa_o))$. For standard commitments we will have $\kappa_c = \kappa_o = \kappa$, and for information theoretically secure commitments we have κ_c is $O(1)$ and κ_o is $O(u)$. So this gives us the complexities we promised in the Introduction.

We show in Section 4 how to construct a secret sharing scheme with the right properties.

Application to Zero-Knowledge Proofs for Circuit Satisfiability An obvious application of our protocol is to give ZK proofs for Boolean circuit satisfiability: the prover commits to the bits on each wire in the circuit C , opens the output as a 1 and shows that, for each AND-gate, the corresponding multiplicative relation holds for the committed bits.

We can apply the Verify Multiplication protocol to do this. Then we have that l is $O(|C|)$. If we run the protocol with error probability 2^{-u} , our expression for the total communication complexity becomes $O(|C|\kappa_c + u(\kappa_c + \kappa_o))$, note that we have to add the cost of committing to the bits in the circuit.

Now we note that if we use the information-theoretic commitments as described above, then $\kappa_c = 1$ and κ_o is $O(u)$. Therefore the complexity is actually $O(|C| + u^2)$ bits, and when dividing by $|C|$ we get communication overhead $O(1)$, as promised in the introduction.

¹ We note already now that since we assume preprocessing for this type of commitment, the simulator constructed for zero-knowledge emulates the verifier's output from the preprocessing as well as his view of the proof (as is standard for set-up models).

4 A Concrete Example

In this section we explain how to design a secret-sharing scheme meeting the demands we stated earlier. For simplicity we first show the details for the case of $u = l$.

As a stepping stone, we consider the following scheme based on Shamir's scheme. Suppose $2(t+l-1) < d$ and $d+l \leq |K|$. Choose pairwise distinct elements $q_1, \dots, q_l, p_1, \dots, p_d \in K$, and define

$$C = \{(f(q_1), \dots, f(q_l), f(p_1), \dots, f(p_d)) \mid f \in K[X]_{\leq t+l-1}\} \subset K^{l+d},$$

where $K[X]_{\leq t+l-1}$ denotes the K -vector space of polynomials with coefficients in K and of degree at most $t+l-1$. Let S correspond to the first l coordinates. Then, by Lagrange Interpolation, it is straightforward to verify that (C, S) is an l -multi-secret K -linear secret sharing scheme of length d , with t -privacy and $(2t+2l-1)$ -product reconstruction. So if we set $t = l$ (and hence the degrees are at most $2l-1$), $d = 8l$, and $|K| \geq 9l$, then $2(t+l-1) = 4l-2 < 8l = d$, $d+l = 9l \leq |K|$, and $\hat{r} = 2t+2l-1 = 4l-1 < 4l = d/2$. In particular, $\frac{\hat{r}-1}{d} < \frac{1}{2}$. Moreover, $e = 2l$, and $\hat{e} = 4l-1$. So all requirements are satisfied, except for the fact that in this approach $|K| = \Omega(\log l)$.

Before we present a scheme which works over a *constant size field*, yet asymptotically it meets all requirements, we describe simple, useful lifting technique. Suppose the finite field of interest K , i.e., the field over which our zero-knowledge problem is defined, does not readily admit the required secret sharing scheme, but that some degree- u extension L of K does. Then we may choose a K -basis of L of the form $1, x, \dots, x^{u-1}$ for some $x \in L$. It is then easy to “lift” the commitment scheme and to obtain one that is L -homomorphic instead: simply consider the elements of L as coordinate-vectors over K , according to the basis selected above, and commit to such a vector by committing separately to each coordinate. This scheme is clearly homomorphic with respect to addition in L . Multiplication by (publicly known) scalars from L is easily seen to correspond to applying an appropriate (publicly known) K -linear form to the vector of K -homomorphic commitments. Furthermore, K is embedded into L by mapping $a \in K$ to $a + 0 \cdot x + \dots + 0 \cdot x^{u-1}$. When committing to $a \in K$, simply commit to a in the original commitment scheme, and append $u-1$ “default commitments to 0.” This way, the protocol problem can be solved over K , with a secret sharing scheme over L . However, communication-wise, even though all further parameters may be satisfied, there are now $O(ul)$ commitments, instead of $O(l)$ as required.

For example, if the above secret sharing scheme is implemented, then since the field K of interest is of constant size, the field L over which the secret sharing is defined must grow proportionally to $\log l$. Hence, the total communication is a logarithmic factor off of what we promised. This is resolved as follows, by using a technique that allows passing to an extension whose degree u is *constant* instead of logarithmic.

Let F be an algebraic function field over the finite field \mathbb{F}_q with q elements. Write g for its genus and n for its number of rational points. Suppose $2g+2(t+l-1) < d$ and $d+l \leq n$. Choose pairwise distinct rational points $Q_1, \dots, Q_l, P_1, \dots, P_d \in F$, and define

$$C = \{(f(Q_1), \dots, f(Q_l), f(P_1), \dots, f(P_d)) \mid f \in \mathcal{L}(G)\} \subset \mathbb{F}_q^{l+d},$$

where G is a divisor of degree $2g+t+l-1$ whose support does not contain any of the Q_j 's nor any of the P_i 's, and where $\mathcal{L}(G)$ is the Riemann-Roch space of G . As before, let S correspond to the first l coordinates. Using a similar result as in [CC06], one proves, using the Riemann-Roch Theorem that (C, S) is an l -multi-secret \mathbb{F}_q -linear secret sharing scheme of length d , with t -privacy and $(2g+2t+2l-1)$ -product reconstruction. Moreover, $e = g+t+l$ and $\hat{e} \leq 3g+2t+2l-1$. Asymptotically, using this result in combination with optimal towers over the *fixed* finite field \mathbb{F}_q where $q \geq 49$ is a square, we get $g/n = \frac{1}{\sqrt{q-1}} < 1/6$. Hence, if we set, for example, $t = l = n/20$ and $d = 19/20n$, then there is $\Omega(l)$ -privacy, $\frac{\hat{r}-1}{d} < c < 1$ for some constant c , and $e = \Omega(l)$, $\hat{e} = \Omega(l)$. Therefore, at most a degree 6 extension of the field of interest is required, as the maximum is attained for $K = \mathbb{F}_2$ with the extension being \mathbb{F}_{64} . Finally, these schemes can be implemented efficiently.

The more general case where u and l are independent parameters follows easily from the above and we leave the details to the reader. The basic reason why it works is that the number of required random

field elements for a sharing (e, \hat{e}) is linear in the required privacy threshold which we want to be $\Theta(u)$ and furthermore the reconstruction threshold (\hat{r}) is linear in the sum of the length of the secret vector and the privacy threshold, which here is $l + \Theta(u)$.

5 A More General Approach

In this section we define linear secret sharing with a more general multiplicative property, and we use the notation from Section 2.2. Let D be an arithmetic circuit over K with v inputs and one output. Then for $\mathbf{c}_1, \dots, \mathbf{c}_v \in K^m$, we define $D(\mathbf{c}_1, \dots, \mathbf{c}_v) \in K^m$ as the vector whose j 'th coordinate is $D((\mathbf{c}_1)_j, \dots, (\mathbf{c}_v)_j)$, i.e., we simply apply D to the j 'th coordinate of all input vectors.

If $C \subset K^m$ is a linear subspace, then C^D is defined as the K -linear subspace generated by all vectors of form $D(\mathbf{c}_1, \dots, \mathbf{c}_v)$ where $\mathbf{c}_1, \dots, \mathbf{c}_v \in C$. Just as for the standard multiplication property, if (C, S) is a secret sharing scheme, then S offers uniformity in C^D , but in general it does not necessarily hold that S^* determines S in C^D . If it does, however, so that (C^D, S) is a linear secret sharing scheme, then we say that (C, S) offers (\tilde{r}, D) -**product reconstruction** if (C^D, S) offers \tilde{r} -product reconstruction.

As a concrete example of this, one may think of Shamir secret sharing. Here, each \mathbf{c}_i is a sequence of evaluations of a polynomial f_i at a fixed set of points. Then $D(\mathbf{c}_1, \dots, \mathbf{c}_v)$ denotes the vector having coordinates of the form $D(f_1(j), \dots, f_v(j))$ for j in the set of evaluation points. These coordinates can be thought as the evaluations of the polynomial $D(f_1, \dots, f_v)$ (defined in the natural way), and sufficiently many of those will determine $D(f_1, \dots, f_v)$ uniquely.

Based on this more general notion, we can design a protocol where a prover commits to vectors $\mathbf{x}_1, \dots, \mathbf{x}_v, \mathbf{z}$ and wants to prove that $D(\mathbf{x}_1, \dots, \mathbf{x}_v) = \mathbf{z}$.

Similarly to what we assumed in the first protocol, we suppose that both the prover and the verifier agreed on using an l -multisecret linear secret sharing scheme (C, S) , for d players, with (\tilde{r}, D) -product reconstruction, and t -privacy. We fix a generator $g : K^{l+e} \rightarrow C$. Moreover, we suppose that $\tilde{g} : K^{l+\tilde{e}} \rightarrow C^D$ is a generator for (C^D, S) and that a public basis for K^{l+e} (respectively for $K^{l+\tilde{e}}$) has been chosen such that the linear mapping g (resp. \tilde{g}) can be computed as the action of a matrix M (resp. \tilde{M}). The protocol goes as follows:

Protocol Verify Circuit

1. The prover chooses v vectors $\mathbf{r}_1, \dots, \mathbf{r}_v \in K^e$, and sets $\rho_j = (\mathbf{x}_j, \mathbf{r}_j)$ for $j = 1, \dots, v$. Define $\mathbf{c}_j = M\rho_j$. Now, the prover computes $\tilde{\rho}_{\mathbf{z}} \in K^{l+\tilde{e}}$ such that $\tilde{\rho}_{\mathbf{z}}$ is consistent with secret \mathbf{z} and such that $\tilde{M}\tilde{\rho}_{\mathbf{z}} = D(\mathbf{x}_1, \dots, \mathbf{x}_v)$. Note that this is possible by solving a system of linear equations, because $D(\mathbf{x}_1, \dots, \mathbf{x}_v) = \mathbf{z}$. We then write $\tilde{\rho}_{\mathbf{z}} = (\mathbf{z}, \tilde{\mathbf{r}}_{\mathbf{z}})$ for some $\tilde{\mathbf{r}}_{\mathbf{z}} \in K^{\tilde{e}}$. Set $\tilde{\mathbf{c}}_{\mathbf{z}} = \tilde{M}\tilde{\rho}_{\mathbf{z}}$.
2. The prover sends vectors of commitments $[\mathbf{r}_j]$, $j = 1, \dots, v$ and $[\tilde{\mathbf{r}}_{\mathbf{z}}]$ to the verifier. Together with the commitments to \mathbf{x}_j and \mathbf{z} , the verifier now holds vectors of commitments $[\rho_j]$, $j = 1, \dots, v$, and $[\tilde{\rho}_{\mathbf{z}}]$.
3. The verifier chooses t uniform indices $O \subset S^*$ and sends them to the prover.
4. Let \mathbf{m}_i be the i 'th row of M and let $\tilde{\mathbf{m}}_i$ be the i 'th row of \tilde{M} . For each $i \in O$, using the homomorphic property of the commitments, both prover and verifier compute commitments

$$[(\mathbf{c}_j)_i] = [\rho_j]^{\mathbf{m}_i}, \text{ for } j = 1, \dots, v, \quad [(\tilde{\mathbf{c}}_{\mathbf{z}})_i] = [\tilde{\rho}_{\mathbf{z}}]^{\tilde{\mathbf{m}}_i}.$$

The prover opens these commitments to the verifier.

5. The verifier accepts if and only if the opened values satisfy $D((\mathbf{c}_1)_i, \dots, (\mathbf{c}_v)_i) = (\tilde{\mathbf{c}}_{\mathbf{z}})_i$ for all $i \in O$.

Using a similar proof as for theorem 1, one easily shows

Theorem 2. *Assume the commitment scheme used is the one described in section 2.1. Then the protocol Verify Circuit is perfect honest-verifier zero-knowledge and if $D(\mathbf{x}_1, \dots, \mathbf{x}_v) \neq \mathbf{z}$, the verifier accepts with probability at most $((\tilde{r} - 1)/d)^t + 1/|L|$.*

The interesting question is whether we can build secret sharing schemes with this type of D -reconstruction and whether the resulting more general protocol offers advantages over the first one.

The answer to the first question is positive, the construction was already hinted at above: we can base a scheme on Shamir secret sharing extended à la Franklin and Yung [FY92] to share blocks of l secrets. This requires polynomials of degree $e = l + t - 1$. Since each multiplication in D doubles the degree of the polynomials, the degree after applying D will be $2^\delta t$ where δ is the multiplicative depth of D . This means that $\tilde{e} = \tilde{r} = 2^\delta t$ for this construction, and d should be a constant factor larger than \tilde{r} to get exponentially small error probability.

We assume for simplicity that the cardinality of K is larger than $d+l$, in order to have the required number of evaluation points. If this is not the case, we can pass to an extension field at cost a logarithmic factor, as explained in the previous section. Note that the algebraic geometric approach presented in Section 2.2 does not give any non-constant improvement over the Shamir-based approach in the setting of D -reconstruction. However, it appears that the algebraic geometric approach can be extended to get a non-trivial improvement here as well, using more advanced techniques.

We can now compare two natural approaches to verifying that committed vectors $\mathbf{x}_1, \dots, \mathbf{x}_v, \mathbf{z}$ satisfy $D(\mathbf{x}_1, \dots, \mathbf{x}_v) = \mathbf{z}$:

The first approach is to perform the Verify Circuit protocol using the secret sharing scheme we sketched. If we go for error probability 2^{-l} and therefore choose t to be $\Theta(l)$, and representing the index set O in the most convenient way (see “Representing a Sequence of Points” in section 3), simple inspection of the protocol shows:

Lemma 1. *Using the Verify Circuit Protocol, the amortized communication complexity to verify one instance of a circuit with multiplicative depth δ and v inputs is $O(2^\delta \kappa + v\kappa + \delta \log l)$ bits for an error probability of 2^{-l} .*

Note that, except for the cost of committing to the inputs the communication complexity only depends on the depth of the circuit.

The second approach is to use the Verify Multiplication protocol. The prover will, for every multiplication gate T in D , commit to a vector \mathbf{z}_T where $(\mathbf{z}_T)_i$ is the output from T in the instance of D where the inputs are $(\mathbf{x}_1)_i, \dots, (\mathbf{x}_v)_i$. Now, for every multiplication gate T the verifier can compute vectors of commitments $[\mathbf{x}_T], [\mathbf{y}_T]$ to the inputs to T (since any linear operations in D “between multiplication gates” can be done by the verifier alone). We then use the Verify Multiplication protocol to check that $\mathbf{x}_T * \mathbf{y}_T = \mathbf{z}_T$. Using this protocol verifying a multiplication has communication cost $O(\kappa)$ bits, so the total cost to verify one instance of the circuit corresponds to $O(\mu\kappa + v\kappa)$ bits, where μ is the number of multiplication gates in D .

Notice that large fan-out comes at no cost in our model, and that linear operations with large fan-in are also for free. Moreover, both approaches generalize easily to circuits with several outputs. Therefore, there is no fixed relation between μ and δ , in particular, we could consider families of circuits where δ is constant or logarithmic in the input size, but μ grows faster than 2^δ . In such a case, using the Verify Circuit protocol is better; it has the interesting property that the amortized cost of verifying a single instance of D can be asymptotically smaller than the number of multiplication gates in D .

In the appendix, we sketch a final variant of the Verify Circuit Protocol using the “MPC in the head approach” [IKOS09] where we try to limit the dominating cost of committing to the \tilde{e} entries of $\tilde{\mathbf{r}}_{\mathbf{z}}$. The idea is as follows: instead of committing to the values in $\tilde{\mathbf{r}}_{\mathbf{z}}$ in the usual way, the prover will simply send the required commitments to shares $[(\tilde{\mathbf{c}}_{\mathbf{z}})_i]$ and use the “MPC in the head” approach to prove to the verifier that the commitments contain the correct shares.

The cost of this approach to generate the required commitments to shares is $O(l^2\kappa + \tilde{e}l^2k)$.

This should be compared to the normal Verify Circuit protocol where the cost of this same step is $O((\tilde{e} + l)\kappa)$. We see that if $\kappa > l^2k$ and $\tilde{e} > l^2$ - which may well be the case in practice - then this solution has smaller cost.

6 Proving Integer Multiplication

In Appendix A, we show a protocol designed for the case where the prover’s secret values are integers. We use a specific integer linear secret sharing scheme based on polynomials and the commitment scheme described in section 2.1. The idea of the protocol is otherwise similar to the one for finite fields. Due to space limitation we only give an informal result here, details can be found in Appendix A.

Theorem 3 (Informal). *Assume the commitment scheme used is the one described in section 2.1. There exists a perfect zero-knowledge proof for showing that committed l -vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ satisfy $\mathbf{x} \cdot \mathbf{y} = \mathbf{z}$, and if for some i , $x_i y_i \neq z_i$, the verifier accepts with probability at most $(2(t+l)/d)^t + O(\kappa/2^\kappa)$.*

7 Commitment Schemes Based on Computational Assumptions

We consider two kinds of commitment schemes: The first one is over a finite field K and can be seen as a function $com_{pk} : K \times H \rightarrow G$ where H, G are finite groups and pk is a public key (this includes the examples suggested in [CD98]). The second one is over the integers, and $com_{pk} : \mathbb{Z} \times \mathbb{Z} \rightarrow G$.

The public key pk is generated by a PPT algorithm \mathcal{G} on input a security parameter κ . To commit to value $x \in K$ or an integer x , the prover chooses r uniformly in H (or, in case of integer commitments, in some appropriate interval) and sends $C = com_{pk}(x, r)$ to the verifier. A commitment is opened by sending x, r . We assume that the scheme is homomorphic, i.e. $com_{pk}(x, r) \cdot com_{pk}(y, s) = com_{pk}(x + y, rs)$. For simplicity, we assume throughout that K is a prime field. Then, by repeated addition, that we also have $com_{pk}(x, r)^y = com_{pk}(xy, r^y)$ for any $y \in K$. We also use $[x]$ as shorthand for a commitment to x in the following, and hence suppress the randomness from the notation.

We consider *computationally hiding* schemes: for any two values x, x' the distributions of $pk, com_{pk}(x, r)$ and $pk, com_{pk}(x', s)$ must be computationally indistinguishable, where pk is generated by \mathcal{G} on input security parameter κ . Such schemes are usually *unconditionally binding*, meaning that for any pk that can be output from \mathcal{G} , there does not exist x, r, x', s with $x \neq x'$ such that $com_{pk}(x, r) = com_{pk}(x', s)$. For such schemes, the prover usually runs \mathcal{G} , sends pk to the verifier and may have to convince him that pk was correctly generated before the scheme is used.

One may also consider *unconditionally hiding and computationally binding* schemes, where $pk, com_{pk}(x, r)$ and $pk, com_{pk}(x', s)$ must be statistically indistinguishable, and where it must be infeasible to find x, r, x', s with $x \neq x'$ such that $com_{pk}(x, r) = com_{pk}(x', s)$.

8 Results with Standard Commitments

Theorem 4. *Assume the commitment scheme used is unconditionally binding and computationally hiding. Then the Verify Multiplication protocol is a computationally honest-verifier zero-knowledge interactive proof system for the language*

$$\left\{ ([x_i], [y_i], [z_i])_{i=1}^l \mid x_i y_i = z_i, \text{ for } i = 1, \dots, l \right\}$$

with soundness error $((\hat{r} - 1)/d)^t$.

In the appendix, we give a proof for the above theorem and explain how to modify the protocol to work for an unconditionally hiding commitment scheme. Using a similar proof as for theorem 4, one easily shows

Theorem 5. *Assume the commitment scheme used is unconditionally binding and computationally hiding. Then the protocol Verify Circuit is a computationally honest-verifier zero-knowledge interactive proof system for the language*

$$\{ ([\mathbf{x}_1], \dots, [\mathbf{x}_v], [\mathbf{z}]) \mid D(\mathbf{x}_1, \dots, \mathbf{x}_v) = \mathbf{z} \}$$

with soundness error $((\tilde{r} - 1)/d)^t$.

Theorem 6 (Informal). *Given an commitment scheme for integers in the interval $[-2^k, \dots, 2^k]$ that is computationally binding and unconditionally hiding. There exists an statistical honest-verifier zero-knowledge argument of knowledge for showing that committed l -vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ satisfy $\mathbf{x} \cdot \mathbf{y} = \mathbf{z}$. The protocol has amortized complexity $O(\kappa + k)$ bits and knowledge error 2^{-l} .*

References

- [BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.
- [BG06] Mihir Bellare and Oded Goldreich. On probabilistic versus deterministic provers in the definition of proofs of knowledge. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(136), 2006.
- [Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, 2000.
- [BSFO11] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. Cryptology ePrint Archive, Report 2011/629, 2011. <http://eprint.iacr.org/>.
- [CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 521–536. Springer, 2006.
- [CCG⁺07] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. Secure computation from random error correcting codes. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2007.
- [CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 1998.
- [CD09] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2009.
- [CDD⁺99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT*, pages 311–326, 1999.
- [CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2000.
- [CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001.
- [DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer, 2002.
- [DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2010.
- [DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.
- [FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997.
- [FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710. ACM, 1992.
- [GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
- [IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
- [KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference*, pages 102–111, 1993.

A Proving Integer Multiplication

In the following, we show a protocol designed for the case where the prover's secret values are integers. We make use of a specific integer linear secret sharing scheme based on polynomials and assume that the underlying commitment scheme is computationally binding and unconditionally hiding. The idea of the protocol is similar to the one for finite fields.

Let

$$\Delta = \prod_{\substack{i,j=1,\dots,d, \\ i \neq j}} (i - j),$$

where d is the number of players. Assume that the secrets s_1, \dots, s_l to be shared satisfy $s_i \in \{-2^k, \dots, 2^k\}$ for all i , for some k . In order to share them, sample random integers $a_1, \dots, a_t \in \{-l2^{k+u}\Delta d!, \dots, l2^{k+u}\Delta d!\}$ (where u is the security parameter) and use Lagrange interpolation over the rationals to find $g \in \mathbb{Q}[X]$ such that

$$g(-i) = s_i \quad \text{and} \quad g(-l - j) = a_j,$$

for $i = 1, \dots, l$ and $j = 1, \dots, t$. Since there are $t + l$ points to interpolate, g has degree (less or) equal to $t + l$. Define $f = \Delta \cdot g$. It follows that f is indeed a polynomial over the integers, since Δ is a multiple of each denominator appearing in the coefficients of g . The shares are then the values $f(1), \dots, f(d)$. Given at least $t + l + 1$ shares, one can reconstruct the secrets, simply by doing Lagrange interpolation over \mathbb{Q} .

Furthermore, any set of at most t shares has distribution statistically independent of s_1, \dots, s_l : let A be an index set designating t players. By Lagrange interpolation we can construct a polynomial $w'_{A,i}$ of degree at most $t + l$ with rational coefficients such that $w'_{A,i}(-i) = 1$ and $w'_{A,i}(j) = 0$ for $i = 1, \dots, l$ and for $j \in A$. From the standard construction of $w'_{A,i}$, it follows that $w_{A,i} = \Delta w'_{A,i}$ has integer coefficients, and that $w_{A,i}(-l - 1), \dots, w_{A,i}(-l - t)$ are all numerically at most $\Delta d!$

Now suppose we have shared the secret consisting of l 0's using a polynomial h . Then $f = h + \sum_{i=1}^l s_i w_{A,i}$ is a polynomial consistent with sharing the secrets s_1, \dots, s_l , but the shares rising from f received by player set A are the same as the ones rising from h . If f is such that $f(-l - 1), \dots, f(-l - t)$ are in the correct interval we conclude that the set of shares in question will be chosen for A with the same probability whether the secrets are $0, \dots, 0$ or s_1, \dots, s_l . But the evaluations $h(-l - 1), \dots, h(-l - t)$ were chosen in an interval a factor 2^u larger than the size of the the evaluations of $\sum_{i=1}^l s_i w_{A,i}$, and hence h and f are both legal, except with probability negligible in u . Hence the distribution of shares seen by A is, for any tuple of secrets, statistically indistinguishable from the distribution for the zeros tuple.

B A Protocol to Prove Integer Multiplication

In this section, we give a protocol allowing prover to show that committed vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with *integer* entries satisfy $\mathbf{x} * \mathbf{y} = \mathbf{z}$ where, consistently with the previous section, an honest prover will choose the committed integers from $\{-2^k, \dots, 2^k\}$. We use the secret sharing scheme from the previous section where we set the security parameter u to be l (this is consistent with previous sections where we have used l as the parameter controlling error probabilities). We use the same notation for commitments as in previous sections, and as an example of the concrete commitment scheme based on computational assumptions, the reader may think of the factoring based scheme from [FO97,DF02]. As an example of the the unconditionally secure commitments scheme, we refer to section 2.1.

Before stating the actual protocol, we fix some notation. Let f be a polynomial of degree equal to m . Write $f(X) = \sum_{j=0}^m f_j X^j$. Define $\mathbf{f} = (f_0, \dots, f_m)$ and $\mathbf{ev}(i) = (1, i, \dots, i^m)$. Notice that

$$f(i) = \sum_{j=0}^m f_j \cdot i^j = \mathbf{f} \cdot \mathbf{ev}(i), \quad \text{and} \quad [f(i)] = \prod_{j=0}^m [f_j]^{i^j} = [\mathbf{f}]^{\mathbf{ev}(i)}.$$

The formulation on the right hand side of these equations is the one used in the protocol, which proceeds as follows. Note that the protocol differs with respect to the kind of commitment scheme used (based on computational assumptions or unconditionally secure).

The prover holds values $\mathbf{x} = (x_1, \dots, x_l)$, $\mathbf{y} = (y_1, \dots, y_l)$ and $\mathbf{z} = (z_1, \dots, z_l)$, has sent commitments $[\mathbf{x}]$, $[\mathbf{y}]$ and $[\mathbf{z}]$ to the verifier. We suppose they both agreed in using the linear secret sharing scheme described in section A. In case they are using a commitment scheme based on computational assumptions, we suppose there exists an interactive zero-knowledge proof of knowledge P_C for the relation

$$C = \{(a, w) \mid a = \text{com}_{pk}(x, r), w = (x, r)\}.$$

Moreover, we assume this interactive proof of knowledge is a Σ -protocol that can prove knowledge of opening for l commitments at once, with knowledge error 2^{-l} . Conversations in such a protocol has form (a, e, z) where e is random challenge issued by the verifier. Because commitments are homomorphic, such a proof of knowledge follows immediately from the techniques described in [CD09]. In the protocol below, we execute a variant of our protocol from the previous sections in parallel with P_C . Thus the overall protocol will have the form of a Σ -protocol which simplifies the proof of soundness.

Note! Text in *italic font* denotes actions performed only if using a commitment scheme based on computational assumptions.

1. The prover chooses $\mathbf{a}_x, \mathbf{a}_y \in \mathbb{Z}^t$ and uses Lagrange interpolation (over the rationals) to generate two polynomials g_x, g_y , having degree $t + l$, such that

$$g_x(-i) = x_i, \quad g_x(-l - j) = (\mathbf{a}_x)_j, \quad g_y(-i) = y_i, \quad g_y(-l - j) = (\mathbf{a}_y)_j,$$

for $i = 1, \dots, l$ and $j = 1, \dots, t$. The prover now sets $\widehat{g}_z = g_x \cdot g_y$, $f_x = \Delta g_x$, $f_y = \Delta g_y$ and $\widehat{f}_z = \Delta^2 \widehat{g}_z$. As explained above, f_x and f_y are polynomials with integral coefficients and have degree at most $t + l$. Notice that \widehat{f}_z is also a polynomial with integral coefficients, but has degree at most $2(t + l)$.

2. The prover sends commitments $[\mathbf{f}_x]$, $[\mathbf{f}_y]$ and $[\widehat{\mathbf{f}}_z]$.
3. The verifier checks that $[\mathbf{x}]$, $[\mathbf{y}]$ and $[\mathbf{z}]$ are consistent with f_x, f_y and \widehat{f}_z : namely for all $i = 1, \dots, l$ it computes

$$[\mathbf{f}_x]^{\text{ev}(-i)}[\Delta x_i]^{-1}, \quad [\mathbf{f}_y]^{\text{ev}(-i)}[\Delta y_i]^{-1}, \quad [\widehat{\mathbf{f}}_z]^{\text{ev}(-i)}[\Delta^2 z_i]^{-1},$$

and asks the prover to open these commitments to zero. If any of these openings do not agree with the commitments, the verifier quits.

4. *The prover defines the vector $x_C = ([\mathbf{x}], [\mathbf{y}], [\mathbf{z}], [\mathbf{f}_x], [\mathbf{f}_y], [\widehat{\mathbf{f}}_z])$ containing committed values. We think of x_C as a vector of instances for the protocol P_C . The prover computes a vector a_C as the first message for the protocol P_C with instance x_C . the prover sends a_C to the verifier.*
5. The verifier chooses t uniform indices $O \subset \{1, \dots, d\}$. Similarly as above, the verifier computes

$$[\mathbf{f}_x]^{\text{ev}(i)} = [(\mathbf{b}_x)_i], \quad [\mathbf{f}_y]^{\text{ev}(i)} = [(\mathbf{b}_y)_i], \quad [\widehat{\mathbf{f}}_z]^{\text{ev}(i)} = [(\widehat{\mathbf{b}}_z)_i],$$

for $i \in O$. The verifier generates a vector e_C as a challenge on (x_C, a_C) according to P_C . The verifier sends e_C together with the index set O to the prover.

6. *The prover computes the vector z_C as a reply for (x_C, a_C, e_C) according to P_C . The prover sends z_C together with the openings of $[(\mathbf{b}_x)_i]$, $[(\mathbf{b}_y)_i]$ and $[(\widehat{\mathbf{b}}_z)_i]$ for $i \in O$.*
7. The verifier accepts if and only if (x_C, a_C, e_C, z_C) is an accepted conversation for P_C and the opened values satisfy $(\mathbf{b}_x)_i \cdot (\mathbf{b}_y)_i = (\widehat{\mathbf{b}}_z)_i$ for $i \in O$.

Using a proof similar to the one of theorem 1 one shows the following result:

Theorem 7. *Assume the commitment scheme used is the one described in section 2.1. Then protocol above is perfect zero-knowledge, and if for some i , $x_i y_i \neq z_i$, the verifier accepts with probability at most $(2(t + l)/d)^t + \kappa/2^\kappa$.*

In the theorem below, we show soundness and honest verifier zero-knowledge for the above protocol. It may seem strange at first sight that the theorem does not assume that commitments are binding. This is because we show that the protocol *unconditionally* is a proof of knowledge that either the prover knows $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with the expected multiplicative relation, or he knows a commitment that he can open in two different ways. Using this result in an application, one would apply the knowledge extractor and then argue that because the commitment scheme is computationally binding, the possibility that the prover breaks the binding property occurs with negligible probability. Since the primary example we know of integer commitments ([FO97,DF02]) has binding based on factoring, applications of this result only need to assume factoring is hard, in contrast to earlier techniques where strong RSA was needed.

Theorem 8. *Assume the homomorphic commitment scheme used is unconditionally hiding. Then the above protocol is a statistical honest-verifier zero-knowledge interactive proof of knowledge for the relation*

$$M = \{(a, w) \mid a = (pk, A_i, B_i, C_i)_{i=1}^l, w = (x_i, r_i, y_i, s_i, z_i, t_i)_{i=1}^l : \\ \text{com}_{pk}(x_i, r_i) = A_i, \text{com}_{pk}(y_i, s_i) = B_i, \text{com}_{pk}(z_i, t_i) = C_i, z_i = x_i y_i \text{ for } i = 1, \dots, l\} \cup \\ \{(a, w) \mid a = (pk, A), w = (v, r, v', r') : \text{com}_{pk}(v, r) = A = \text{com}_{pk}(v', r'), v \neq v'\}$$

with knowledge error $ke_M = \max\{(2(t+l)/d)^t, 2^{-l}\}$.

On the complexity of the protocol We now examine the complexity of the integer multiplication protocol assuming we want a knowledge error that is exponentially small in l , as in previous sections. It is easy to see that this can be arranged if we choose the parameters t and d to be $\Theta(l)$. Recall also that we already chose the statistical security parameter of the secret sharing scheme to be $\Theta(l)$. With these parameter choices, simple inspection of the protocol and secret sharing scheme shows that the amortized complexity per multiplication proved is $O(\kappa + k)$. This also includes the cost of the proof P_C : This can be verified by a direct inspection of the technique from [CD98], for a case where a proof is given for l commitments in parallel and where the statistical security parameter of the proof is also set to l .

C Verify Multiplication Protocol for Unconditionally Hiding Commitments

We briefly sketch how to modify the protocol to work for an unconditionally hiding and computationally binding commitment scheme. The protocol would then be a proof of knowledge that the prover can open his input commitments to reveal strings $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with $\mathbf{x} * \mathbf{y} = \mathbf{z}$. We need to add in Step 2 that the prover must prove that he knows how to open all the commitments $[\mathbf{x}], [\mathbf{y}], [\mathbf{z}], [\mathbf{r}_x], [\mathbf{r}_y], [\widehat{\mathbf{r}}_z]$. This can be done by simply invoking the amortized efficient zero-knowledge protocol from [CD09] since the commitment function we assume is exactly of the form this protocol can handle. The overhead introduced by this is only a constant factor.

The proof of zero-knowledge is exactly the same, except that we get perfect (statistical) zero-knowledge if the commitment scheme is perfect (statistically) hiding.

For soundness, we argue that parameters are chosen such that $((\widehat{r} - 1)/d)^t$ is negligible in the security parameter, and if the prover convinces the verifier with non-negligible probability, then there exists a knowledge extractor that uses the prover to compute openings of his input commitments to strings $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with $\mathbf{x} * \mathbf{y} = \mathbf{z}$ (except with negligible probability). This algorithm would first invoke the knowledge extractor for the protocol from [CD09] to get opening of all the prover's initial commitments, to strings $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}_x, \mathbf{r}_y, \widehat{\mathbf{r}}_z$. We claim that except with negligible probability, we will have $\mathbf{x} * \mathbf{y} = \mathbf{z}$.

This follows since, as we now argue, if $\mathbf{x} * \mathbf{y} \neq \mathbf{z}$ then we could break the binding property of the commitments. To see this, notice that from the openings we know of the prover's initial commitments, we can use the homomorphic property to compute openings of any commitment to a share that the prover can be asked to open in Step 4. Call these the *predetermined* openings. Note that the shares in question will be consistent with secret sharing the strings $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Now we send a random challenge to the prover, and by assumption on the prover, his reply will pass the verifier’s test with non-negligible probability, i.e., for all $i \in O$, the opened values $s_{x,i}, s_{y,i}, s_{z,i}$ will satisfy $s_{x,i}s_{y,i} = s_{z,i}$. However, this is not the case for the predetermined openings of the same commitments: it immediately follows from the soundness proof of the previous theorem that because the predetermined openings are consistent with actually secret sharing $\mathbf{x}, \mathbf{y}, \mathbf{z}$, these openings will satisfy the multiplicative relation with only negligible probability $((\hat{r}-1)/d)^t$ (over the choice of the verifier’s challenge). It follows that with non-negligible probability, there will be at least one commitment to a share for which the predetermined opening is different from the opening done by the prover in response to the challenge. We have therefore broken the binding property.

D Using MPC in the Head for the Verify Circuit Protocol

We now sketch a final variant of the Verify Circuit protocol that leads to a complexity that is in general incomparable to the first one, but for reasonable parameter values will give an improvement.

The idea is as follows: instead of committing to the values in $\tilde{\mathbf{r}}_{\mathbf{z}}$ in the usual way, the prover will simply send the required commitments to shares $[(\tilde{\mathbf{c}}_{\mathbf{z}})_i]$ and use the “MPC in the head” approach from [IKOS09] (the IKOS compiler) to prove to the verifier that the commitments contain the correct shares.

To use this approach, one first specifies a multiparty protocol that creates the desired output, the IKOS compiler will then produce a 2 party zero-knowledge protocol proving the result is correct, assuming also a suitable commitment scheme (not necessarily the one we use in the basic protocol).

The multiparty protocol goes as follows: we will have $a \in \Theta(l)$ players, of which a constant fraction may be actively corrupted. The first step is to generate a set of random secret shared values $r_1, \dots, r_{\tilde{e}}$, shared among the a players using standard Shamir sharing over K . Using a simple variant of the protocol by Hirt and Berliova based on hyperinvertible matrices, this can be done in total communication complexity $O(\tilde{e}lk)$ bits where k is the size of a field element. We now set $\tilde{\mathbf{r}}_{\mathbf{z}} = (r_1, \dots, r_{\tilde{e}})$ and we let the shares of these values be $r_{u,j}$, $u = 1, \dots, \tilde{e}$, $j = 1, \dots, a$. Let \mathbf{n}_i be the last \tilde{e} entries of \mathbf{m}_i . Then each player outputs a commitment to the inner product $[t_{i,j}] = [(r_{1,j}, \dots, r_{\tilde{e},j}) \cdot \mathbf{n}_i]$.

Let $\lambda_1, \dots, \lambda_a$ be the Lagrange coefficients to reconstruct the secret given correct shares. Everybody can now compute $[t_i] = \prod_j [t_{i,j}]^{\lambda_j}$.

Note that we do not yet know if the value is correct, but if all virtual players output correct commitments, then the desired commitment to $(\tilde{\mathbf{c}}_{\mathbf{z}})_i$ can be computed as a “linear combination” of the commitments \mathbf{z} and $[t_i]$.

Note that the $t_{i,j}$ ’s are in fact Shamir shares in t_i , and they are all correct, except for a constant fraction. Therefore, it follows that t_i is correct if all $t_{i,j}$ are on a polynomial of low enough degree. We check this by computing commitments to the “syndrome” of the set of $t_{i,j}$ ’s: these commitments should contain all 0’s. In a normal multiparty situation, we could not open these commitments, but in our case a prover is executing the protocol in his head, so we can just ask the prover to open these.

When we compile this protocol to a 2-party protocol, the idea is, as mentioned, that the prover executes the protocol in his head and commits to the views of all players. We do this with a separate commitment scheme that does not need to be homomorphic. The verifier asks the prover to open the views of a randomly chosen unqualified subset of players and checks the views for consistency. The IKOS results show that if the protocol has not worked correctly, the verifier will reject, except with probability $2^{-\Theta(a)}$. As a result, we get the commitments to shares we wanted.

Since a and t , the number of opened shares are $\Theta(l)$, the cost of this is $O(l^2\kappa)$ bits for the commitments and $\tilde{e}l^2k$ bits for the views of virtual players.

This new protocol should be compared to the normal one where the cost is $O((\tilde{e} + l)\kappa)$. We see that if $\kappa > l^2k$ and $\tilde{e} > l^2$ then the new solution has smaller cost.

E Proofs

Proof (Theorem 1). For soundness, we suppose that the prover is dishonest (so $x_i y_i \neq z_i$ for some i) and we compute the probability that the protocol accepts. Note first that, from the prover’s commitments, vectors $\mathbf{c}_x, \mathbf{c}_y, \widehat{\mathbf{c}}_z$ are determined, where we know that \mathbf{x}, \mathbf{y} and \mathbf{z} respectively appear in coordinates designated by S . Since $x_i y_i \neq z_i$ for some i , it follows that $\mathbf{c}_x * \mathbf{c}_y \neq \widehat{\mathbf{c}}_z$.

Denote by $T \subset S^*$ the index set in the share space where the vectors $\mathbf{c}_x * \mathbf{c}_y$ and $\widehat{\mathbf{c}}_z$ agree.

Note that the cardinality of T is at most $\widehat{r} - 1$, because $\mathbf{c}_x * \mathbf{c}_y$ and $\widehat{\mathbf{c}}_z$ are consistent with different secrets. In order for the prover to be successful, one of the following must hold:

- All t entries the verifier asks the prover to unveil are in T .
In this case the prover can behave honestly, since the choice of the verifier points to values that already satisfy the expected multiplicative relation.
- The prover can break the binding property of the commitment scheme.
In this case the prover can open commitments to arbitrary values.

For the first event: the probability that one entry chosen by the verifier is in T is at most equal to $(\widehat{r} - 1)/d$, since the choice is uniform. Repeating this argument t times implies that the event that all t entries asked by the verifier are in T happens with probability equal to $((\widehat{r} - 1)/d)^t$.

If this event does not happen, there is at least an entry chosen by the verifier that lies outside T . The prover can still succeed the check by opening to values that satisfy the multiplicative relation expected by the verifier, even for those entries that lie outside T . Since there is at least one entry outside T , the prover must break the binding property of the commitment scheme to succeed, and this event happens with probability $1/|L|$, as shown in section 2.1. Therefore, the soundness error of the protocol is bounded by $(\widehat{r} - 1)/d + 1/|L|$.

To show zero-knowledge, we build the following simulator.

- For the setup, the simulator runs a local copy of the ideal functionality for the preprocessing, obtaining values $(u_1, m_{u_1}), \dots, (u_i, m_{u_i}), \dots \in K \times L$ and values $(a_{u_1}, b_{u_1}), \dots, (a_{u_i}, b_{u_i}), \dots \in L \times L$ such that

$$m_{u_i} = a_{u_i} \cdot u_i + b_{u_i}, \quad \text{for all } i.$$

The simulator sends the values $(a_{u_1}, b_{u_1}), \dots, (a_{u_i}, b_{u_i}), \dots$ to the verifier.

- The simulator runs the protocol exactly as a honest prover until step 3.
- In step 4, in order to open each value $[(\mathbf{c}_x)_i] = [\rho_x]^{m_i}$ (respectively $[(\mathbf{c}_y)_i]$), the simulator first computes the corresponding key $(a, b_{(\mathbf{c}_x)_i})$ (respectively $(a, b_{(\mathbf{c}_y)_i})$), samples a uniform $x'_i \in K$ (respectively y'_i), computes $m_{x'_i} = a \cdot x'_i + b_{(\mathbf{c}_x)_i}$ (respectively $m_{y'_i} = a \cdot y'_i + b_{(\mathbf{c}_y)_i}$) and sends $(x'_i, m_{x'_i})$ (respectively $(y'_i, m_{y'_i})$) as opening. To open $[(\widehat{\mathbf{c}}_z)_i]$ the simulator computes the corresponding key $(a, b_{(\widehat{\mathbf{c}}_z)_i})$, sets $z'_i = x'_i \cdot y'_i$, computes $m_{z'_i} = a \cdot z'_i + b_{(\widehat{\mathbf{c}}_z)_i}$ and sends $(z'_i, m_{z'_i})$ as opening.

The simulation is clearly polynomial time. The distribution of the messages sent to the verifier is the same as in a real execution of the protocol until step 3. In step 4 the opened values have the same distribution as in the real protocol (x'_i, y'_i are uniform, and $z'_i = x'_i \cdot y'_i$), and the openings are valid, since they pairs sent satisfy the linear relation expected by the verifier. \square

Proof (Theorem 4). For soundness, one can follow the lines of the proof of theorem 1.

To show (honest-verifier) zero-knowledge, the idea is to “execute the protocol” exactly as the honest prover would have done, but assuming that all secret values are 0. After that, we adjust the relevant values so they become consistent with the actual values of \mathbf{x}, \mathbf{y} and \mathbf{z} .

So we first generate random vectors $\rho_0^x = (\mathbf{0}, \mathbf{r}_0^x), \rho_0^y = (\mathbf{0}, \mathbf{r}_0^y)$, both consistent with sharing the all-0 vector. We compute $\widehat{\rho}_0^z = (\mathbf{0}, \mathbf{r}_0^z)$ such that $\widehat{M} \widehat{\rho}_0^z = (M \rho_0^x) * (M \rho_0^y)$. We then choose a random subset $A \subset S^*$ of t indices. Note that we have $(M \rho_0^x)_i (M \rho_0^y)_i = (\widehat{M} \widehat{\rho}_0^z)_i$ for $i \in A$, and that these shares have the same distribution as in the real conversation, since any t shares have distribution independent of the actual secrets. We then form random vectors of commitments $[\mathbf{r}_0^x], [\mathbf{r}_0^y], [\mathbf{r}_0^z]$. Note that since the commitment function is a

homomorphism from $K \times H$ to G , the neutral element 1_G is a commitment to $0 \in K$. Therefore we can form vectors of commitments as follows

$$[\rho_{\mathbf{0}}^{\mathbf{x}}] = ((1_G, \dots, 1_G), [\mathbf{r}_{\mathbf{0}}^{\mathbf{x}}]), \quad [\rho_{\mathbf{0}}^{\mathbf{y}}] = ((1_G, \dots, 1_G), [\mathbf{r}_{\mathbf{0}}^{\mathbf{y}}]), \quad [\widehat{\rho}_{\mathbf{0}}^{\mathbf{z}}] = ((1_G, \dots, 1_G), [\mathbf{r}_{\mathbf{0}}^{\mathbf{z}}]).$$

As described above, we can assume existence of sweeping vectors $\mathbf{w}_{A,j}$ and $\widehat{\mathbf{w}}_{A,j}$ for the secret sharing schemes (C, S) and (\widehat{C}, S) , respectively, and we know that the vectors

$$\rho_{\mathbf{x}} = \rho_{\mathbf{0}}^{\mathbf{x}} + \sum_{j=1}^l x_j \cdot \mathbf{w}_{A,j}, \quad \rho_{\mathbf{y}} = \rho_{\mathbf{0}}^{\mathbf{y}} + \sum_{j=1}^l y_j \cdot \mathbf{w}_{A,j}, \quad \widehat{\rho}_{\mathbf{z}} = \widehat{\rho}_{\mathbf{0}}^{\mathbf{z}} + \sum_{j=1}^l z_j \cdot \widehat{\mathbf{w}}_{A,j}$$

are consistent with sharing \mathbf{x}, \mathbf{y} and \mathbf{z} , respectively, but where the subset A gets the same shares as when 0 's were shared. The simulator cannot compute $\rho_{\mathbf{x}}, \rho_{\mathbf{y}}, \widehat{\rho}_{\mathbf{z}}$, but it can compute commitments to them. Using the fact that the commitments $[x_j], [y_j], [z_j]$ are given and the sweeping vectors are public, it can compute, for instance, a vector of commitments $[x_i \cdot \mathbf{w}_{A,j}]$ and hence

$$[\rho_{\mathbf{x}}] = [\rho_{\mathbf{0}}^{\mathbf{x}}] \prod_{j=1}^l [x_j \cdot \mathbf{w}_{A,j}], \quad [\rho_{\mathbf{y}}] = [\rho_{\mathbf{0}}^{\mathbf{y}}] \prod_{j=1}^l [y_j \cdot \mathbf{w}_{A,j}], \quad [\widehat{\rho}_{\mathbf{z}}] = [\widehat{\rho}_{\mathbf{0}}^{\mathbf{z}}] \prod_{j=1}^l [z_j \cdot \widehat{\mathbf{w}}_{A,j}].$$

It is easy to verify that because we used neutral elements 1_G as the first entries in $[\rho_{\mathbf{0}}^{\mathbf{x}}], [\rho_{\mathbf{0}}^{\mathbf{y}}], [\widehat{\rho}_{\mathbf{0}}^{\mathbf{z}}]$, the first l entries of $[\rho_{\mathbf{x}}], [\rho_{\mathbf{y}}], [\widehat{\rho}_{\mathbf{z}}]$ as computed above will exactly be $[\mathbf{x}], [\mathbf{y}]$ and $[\mathbf{z}]$. The simulator therefore extracts the last e commitments from $[\rho_{\mathbf{x}}]$ and $[\rho_{\mathbf{y}}]$, and the last \widehat{e} commitments from $[\widehat{\rho}_{\mathbf{z}}]$, and uses these to simulate the commitments sent in Step 2.

It then outputs the index set A as simulation of step 3.

For step 4, note that the simulator may compute and open commitments to

$$[(M\rho_{\mathbf{x}})_i] = [(M\rho_{\mathbf{0}}^{\mathbf{x}})_i], \quad [(M\rho_{\mathbf{y}})_i] = [(M\rho_{\mathbf{0}}^{\mathbf{y}})_i], \quad [(\widehat{M}\widehat{\rho}_{\mathbf{z}})_i],$$

for $i \in A$, where these equalities follow by the sweeping vector properties. By construction, the opened values satisfy the multiplicative property expected by the verifier.

This simulation is clearly polynomial time, and we argued underway that the distribution of all values that are opened are exactly as in a real conversation. The commitments $[\rho_{\mathbf{x}}]$ and $[\rho_{\mathbf{y}}]$ are also distributed correctly. Therefore, the only difference between simulation and conversation lies in the distribution of $\widehat{\rho}_{\mathbf{z}}$ hidden in $[\widehat{\rho}_{\mathbf{z}}]$ (in a real conversation, the choice of $\widehat{\rho}_{\mathbf{z}}$ ensures that the resulting $\widehat{\mathbf{c}}_{\mathbf{z}}$ satisfies $(\mathbf{c}_{\mathbf{x}})_i (\mathbf{c}_{\mathbf{y}})_i = (\widehat{\mathbf{c}}_{\mathbf{z}})_i$ for all indices i , whereas for the simulation this only holds for $i \in A$). It therefore follows from the hiding property of commitments and a standard hybrid argument that simulation is computationally indistinguishable from real conversations. \square

Proof (Theorem 8). For soundness, for any prover P^* that makes the protocol accept with probability p we build a knowledge extractor E_M having running time $(p - ke_M)^{-1} \text{poly}(u)$, where ke_M is equal to $\max\{(2(t+l)/d)^t, 2^{-l}\}$. The latter equality allows us to assume $p > (2(t+l)/d)^t$. Note that by the result from [BG06], we may assume that P^* is deterministic. Therefore p is simply the fraction of challenges e_C, O that P^* answers correctly.

- (i) E_M runs the protocol with P^* until step 3; E_M stores each opening (v, r) in a list L .
- (ii) E_M continues the protocol. It receives a_C during step 4.
- (iii) E_M sends e_C, O computed according to the protocol at step 5.
- (iv) During step 6 E_M receives z_C and the openings of $[(\mathbf{b}_{\mathbf{x}})_i], [(\mathbf{b}_{\mathbf{y}})_i]$ and $[(\widehat{\mathbf{b}}_{\mathbf{z}})_i]$ for $i \in O$. E_M rewinds the prover to step 5 and goes to (iii) until it sees two conversations $(x_C, a_C, e_C, z_C), (x_C, a_C, e'_C, z'_C)$ valid for P_C and such that $e_C \neq e'_C$. At this point E_M can retrieve the witness for x_C , namely the values and the randomness used to make the commitments.
- (v) E_M checks whether $\mathbf{x} * \mathbf{y} = \mathbf{z}$. If that is the case, it outputs $w = (x_i, y_i, z_i, r_i, s_i, t_i)_{i=1}^l$ as a witness for the committed values $[\mathbf{x}], [\mathbf{y}]$ and $[\mathbf{z}]$ and quits.

- (vi) E_M performs the check of step 3 on its own, using the retrieved values and randomness. Each result (v', r') is stored on a list L' , using the same ordering as the one for L (i.e. for each possible j , the j -th entry of L and of L' correspond to the opening of the same commitment). If there exists an index j such that $L_j = (v, r)$, $L'_j = (v', r')$ and $v \neq v'$, then E_M outputs $w = (v, r, v', r')$ as a witness for $\text{com}_{pk}(v, r) = \text{com}_{pk}(v', r')$ and quits.
- (vii) E_M defines T as $i \in T$ if and only if $x_i y_i = z_i$. Then, E_M rewinds the prover to step 5 and sends e_C, O according to the protocol.
- (viii) During step 6, if for some index $i \notin T$ the prover outputs $(x'_i, r'_i, y'_i, s'_i, z'_i, t'_i)$ such that $x'_i y'_i = z'_i$, then E_M outputs $w = (x_i, r_i, x'_i, r'_i)$ if $x_i \neq x'_i$, $w = (y_i, s_i, y'_i, s'_i)$ if $y_i \neq y'_i$, or $w = (z_i, t_i, z'_i, t'_i)$ if $z_i \neq z'_i$ and quits. Else E_M rewinds the prover to step 5 and repeats this step.

The expected running time of this algorithm can be analyzed as follows:

- Step (i) runs in polynomial time, since E_M stores a polynomial amount of data. Notice that the check at step 3 must pass, since p is bigger than zero.
- Step (ii), (iii), (iv) run in polynomial time. The number of rewinds to pass step (iv) is bounded by $2(p - 2^{-l})^{-1}$, which is under the constrains. Retrieving the commitments requires polynomial time (from the special soundness property of sigma protocols).
- Step (v) runs in polynomial time (l multiplications).
- Step (vi) runs in polynomial time, since it requires to perform a polynomial amount of linear operations and multiplications.
- Step (vii) runs in polynomial time.
- Step (viii) happens if P^* makes the test $\mathbf{x} * \mathbf{y} = \mathbf{z}$ fail. We now bound the probability p that P^* succeeds in the protocol in such a situation. In order for P^* to be successful, it had to open the values pointed by O correctly. Since $\mathbf{x} * \mathbf{y} \neq \mathbf{z}$, then $f_x f_y \neq \hat{f}_z$. Notice that $f_x f_x$ and \hat{f}_z are both polynomials of degree $2(t + l)$ and since they are distinct, they have at most $2(t + l)$ roots in common. This implies that one way for P^* to succeed is that all the entries in O point to common roots (that is, $O \subset T$). Since the choice of O is uniform and independent from P^* 's choices, the probability that $O \subset T$ is $(2(t + l)/d)^t$. Since p is assumed to be greater than $(2(t + l)/d)^t$, it means there exists some set $O \not\subset T$ that make P^* succeed. The probability for a uniform O to make P^* succeed and $O \not\subset T$ is equal to $p - (2(t + l)/d)^t$. This implies that the expected number of rewinds in step (viii) is equal to $(p - (2(t + l)/d)^t)^{-1}$, so the total running time of the algorithm is within the constrains even if it terminates in step (viii).

To show (honest-verifier) zero-knowledge we use the same technique we exploit in the field scenario. The simulator samples two random polynomials h_x, h_y of degree $t + l$ such that $h_x(-i) = 0 = h_y(-i)$, that is h_x, h_y are both consistent with sharing the secret consisting of l zeros. It then computes $\hat{h}_z = h_x h_y$. Let $A \subset \{1, \dots, d\}$ be a subset of players of size t . Notice that $h_x(i)h_y(i) = \hat{h}_z(i)$ for all $i \in A$ and that these shares have distribution statistically indistinguishable from a real conversation, since any t shares are essentially independent of the actual secrets. Using the polynomials $w_{A,i}$, $i = 1, \dots, l$ we define

$$f_x = h_x + \sum_{i=1}^l x_i w_{A,i}, \quad f_y = h_y + \sum_{i=1}^l y_i w_{A,i}, \quad \hat{f}_z = \hat{h}_z + \sum_{i=1}^l \Delta z_i w_{A,i}.$$

These three polynomials are consistent with sharing \mathbf{x}, \mathbf{y} and \mathbf{z} , but where the subset A gets the same shares as when 0's were shared. The simulator cannot compute these polynomials, but it can compute commitments to the coefficients. Using the fact that the commitments $[\mathbf{x}], [\mathbf{y}]$ and $[\mathbf{z}]$ are given, and the polynomials $w_{A,i}$ are public, it can compute commitments

$$[\mathbf{f}_x] = [\mathbf{h}_x] \prod_{k=1}^l [x_k \cdot \mathbf{w}_{A,k}], \quad [\mathbf{f}_y] = [\mathbf{h}_y] \prod_{k=1}^l [y_k \cdot \mathbf{w}_{A,k}], \quad [\hat{\mathbf{f}}_z] = [\hat{\mathbf{h}}_z] \prod_{k=1}^l [z_k \cdot \Delta \mathbf{w}_{A,k}].$$

Step 2 is simulated sending commitments $[\mathbf{f}_x], [\mathbf{f}_y]$ and $[\hat{\mathbf{f}}_z]$. The verifier in step 3 checks the consistency of the received data and the check passes. We here prove it for x_i (with a similar proof one shows the check passes

for y_i, z_i , for $i = 1, \dots, l$). The verifier can use the homomorphic properties of the commitment schemes to check whether

$$[\mathbf{f}_x]^{\text{ev}(-i)} \cdot [\Delta x_i]^{-1} = [0].$$

From the construction of f_x , it follows that

$$\begin{aligned} [\mathbf{f}_x]^{\text{ev}(-i)} \cdot [\Delta x_i]^{-1} &= \left([\mathbf{h}_x]^{\text{ev}(-i)} \cdot \prod_{k=1}^l [x_k \cdot \mathbf{w}_{\mathbf{A},k}]^{\text{ev}(-i)} \right) \cdot [\Delta x_i]^{-1} \\ &= [h_x(-i)] \cdot \prod_{k=1}^l [x_k \cdot w_{A,k}(-i)] \cdot [\Delta x_i]^{-1} \\ &= [0] \cdot [\Delta x_i] \cdot [\Delta x_i]^{-1} = [0]. \end{aligned}$$

For step 4, the simulator can compute and open commitments

$$[(\mathbf{f}_x)_i] = [(\mathbf{h}_x)_i], \quad [(\mathbf{f}_y)_i] = [(\mathbf{h}_y)_i], \quad [(\widehat{\mathbf{f}}_z)_i] = [(\widehat{\mathbf{h}}_z)_i],$$

for $i \in A$. By construction, the opened values satisfy the multiplicative property expected by the verifier.

This simulation is clearly polynomial time, and we argued underway that the distribution of all values that are opened are statistically close to those of a real conversation. The commitments $[\mathbf{f}_x]$ and $[\mathbf{f}_y]$ are also distributed correctly. Therefore, the only difference between simulation and conversation lies in the distribution of $\widehat{\mathbf{f}}_z$ hidden in $[\widehat{\mathbf{f}}_z]$ (in a real conversation, the choice of $\widehat{\mathbf{f}}_z$ ensures that the resulting $\widehat{\mathbf{b}}_z$ satisfies $(\mathbf{b}_x)_i (\mathbf{b}_y)_i = (\widehat{\mathbf{b}}_z)_i$ for all indices i , whereas for the simulation this only holds for $i \in A$). Since commitments are statistically hiding, it follows that the simulation is computationally indistinguishable from a real conversation. \square